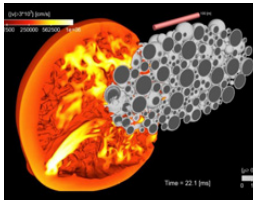
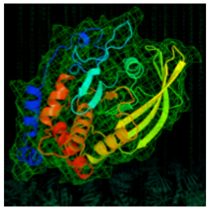
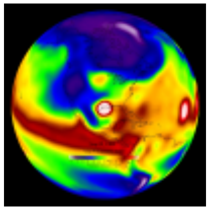
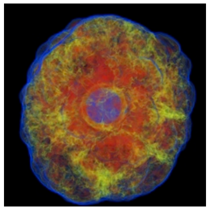
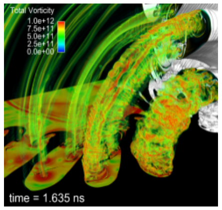
Trinity Platform Introduction and Usage Model

ACES Team

August 2015

LA-UR-15-26834

Topics

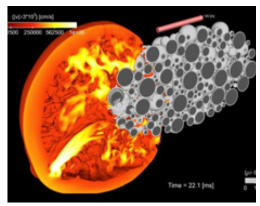
- Introduction to Sierra (LLNL) - NDA
- Trinity status and time-line
- Trinity platform update
- Introduction to the Trinity Usage Model
- User questions/feedback

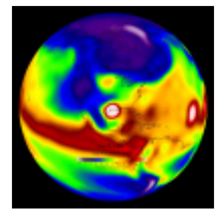
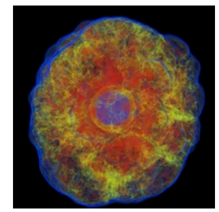
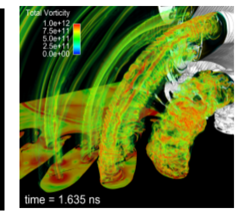
Detailed Agenda

Time	Topic
5	Meeting introduction & goals (Manuel Vigil)
20	Sierra Introduction, NDA-Lab MOWs only (Robin Goldstone, LLNL)
Trinity Platform & Project Status	
10	Trinity project timeline w/ ATCC process (Manuel Vigil)
10	Trinity platform overview & status including classified/unclassified resources (Scott Hemmert)
10	File systems (Brett Kettering, Ruth Klundt)
Trinity Usage Model	
5	Overview of Trinity Usage Model (Jeff Johnson, Karen Haskell)
10	User support and Getting Started – Accessing, help, logging in (Ben Santos, Rob Cunningham, Lisa Ice)
10	Programming environment (Jennifer Green, David Shrader, Kevin Pedretti, Mahesh Rajan)
15	Burst Buffer (Cornell Wright)

Time	Topic
10	**Break**
10	Application Readiness (Cornell Wright, Joel Stevenson)
15	Center of Excellence (Tim Kelley (LANL), Hai Ah Nam (LANL), Rob Hoekstra(SNL), Mike Glass (SNL), (LLNL))
10	Campaign storage (Kyle Lamb, Brett Kettering)
10	Visualization & Data analysis (Laura Monroe, David Karelitz, Warren Hunt)
10	Data Movement & Archive (Brett Hollander, Susie McRee)
30	**User forum & Trinity questions (all speakers)**
5	Wrap up (Manuel Vigil)

Trinity Usage Model - Timeline



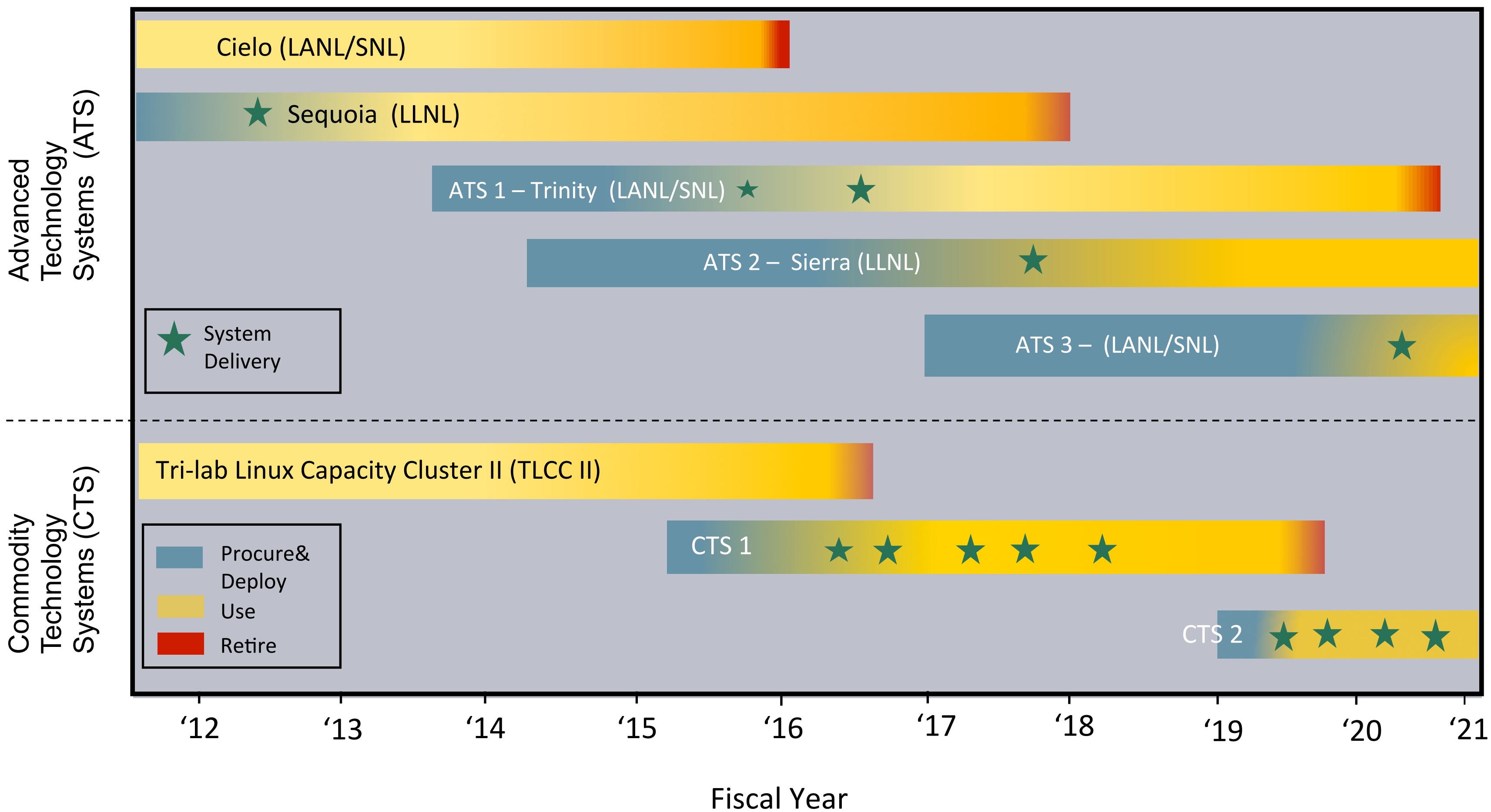




Manuel Vigil

August 2015

LA-UR-15-26470

ASC Platform Timeline



ATS bars (start represents CD-1 signed off, RFP released, and program budget allocations)
CTS bars (start represents RFP released and program budget allocations)

Jan. 2015

Overview of Trinity Award

- Subcontractor
  - Cray, Inc.
- Firm Fixed Price Subcontract:
  - Trinity Platform (including File System)
  - Burst Buffer
  - 2 Application Regression Test Systems
  - 1 System Development Test System
  - On-site System and Application Analysts
  - Center of Excellence for Application Transition Support
  - Advanced Power Management
  - Trinity System Maintenance
- Acquired and managed by the New Mexico Alliance for Computing at Extreme Scale

Trinity Project Drivers

- Satisfy the mission need for more capable platforms
  - Trinity is designed to support the largest, most demanding ASC applications
  - Increases in geometric and physics fidelities while satisfying analysts' time-to-solution expectations
  - Foster a competitive environment and influence next generation architectures in the HPC industry
- Trinity is enabling new architecture features in a production computing environment
  - Trinity's architecture will introduce new challenges for code teams: transition from multi-core to many-core, high-speed on-chip memory subsystem, wider SIMD/vector units
  - Tightly coupled solid state storage serves as a "burst buffer" for checkpoint/restart file I/O & data analytics, enabling improved time-to- solution efficiencies
  - Advanced power management features enable measurement and control at the system, node, and component levels, allowing exploration of application performance/watt and reducing total cost of ownership
- Mission Need Requirements are primarily driving memory capacity
  - Over 2 PB of aggregate main memory

Trinity Platform

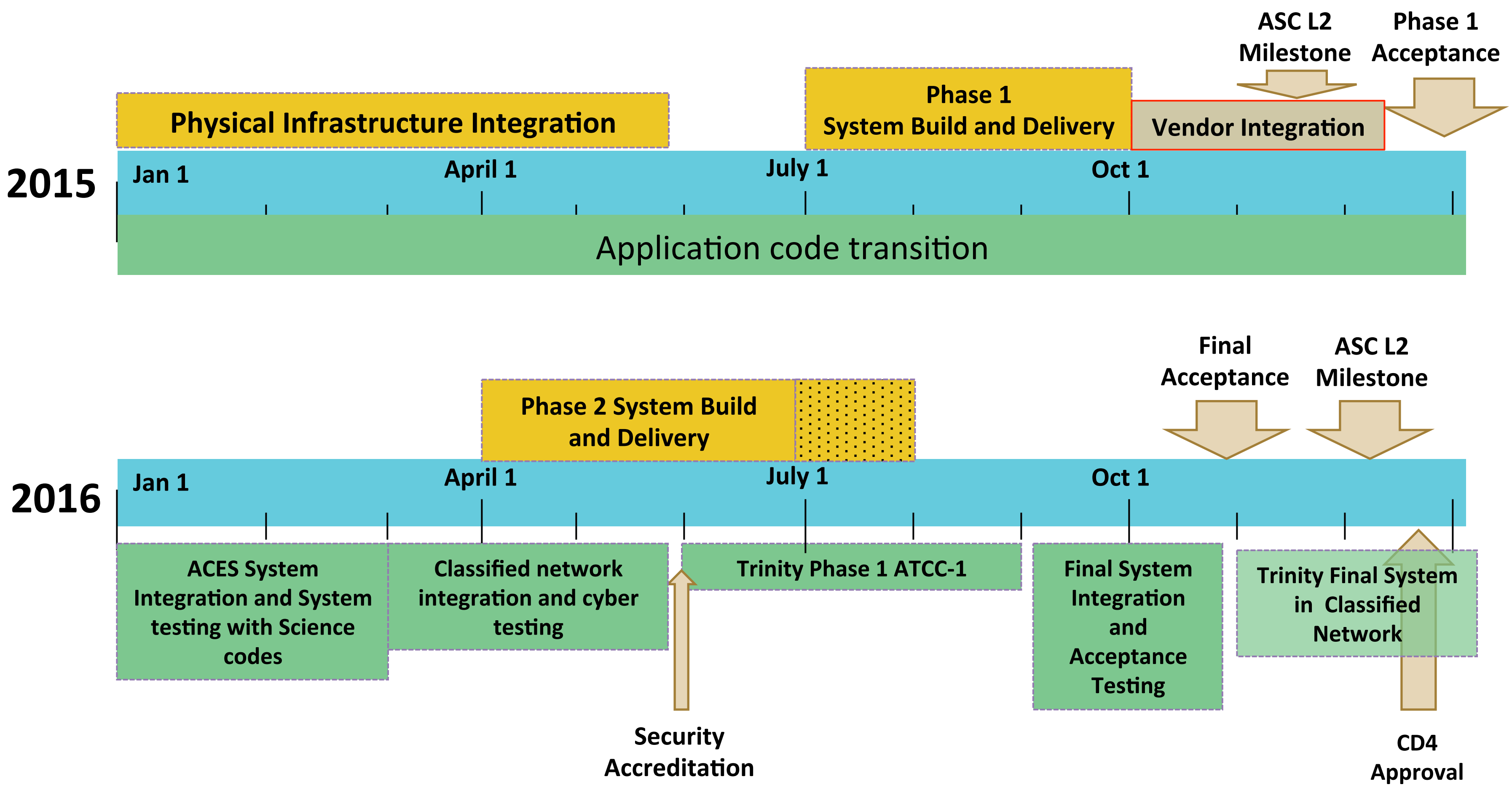
- Trinity is a single system that contains both Intel Haswell and Knights Landing processors
  - Haswell partition satisfies FY16 mission needs (well suited to existing codes).
  - KNL partition delivered in FY16 results in a system significantly more capable than current platforms and provides the application developers with an attractive next-generation target (and significant challenges)
  - Aries interconnect with the Dragonfly network topology

- Based on mature Cray XC30 architecture with Trinity introducing new architectural features
  - Intel Knights Landing (KNL) processors
  - Burst Buffer storage nodes
  - Advanced power management system software enhancements

Advanced Technology System Scheduling Governance Model
Advanced Technology Computing Campaigns (ATCCs)

- The goal of this model is to effectively allocate and schedule AT computing resources among all three NNSA laboratories for weapons deliverables that merit priority on this class of resource
- Objectives of this governance model are to:
  - Ensure that AT system resources are allocated on a priority-driven basis according to SSP Program requirements;
  - Utilize ASC AT systems for the most demanding workload categories, for which they were designed and procured; and
  - Support the role of AT systems to prepare ASC resources (including our people, our applications, and our computing environments), for significant changes in *future* computer and system architectures.
- Allocations
  - Each Lab has 1/3 allocation on each AT system and will hold an internal process for each six-month campaign to review/prioritize proposed projects.
- Reporting and Accountability
  - Host labs will prepare ATCC utilization reports on a monthly basis.
  - Once a year the host lab briefs NNSA HQ staff on the simulations that were carried out in the previous two ATCCs on the AT system.

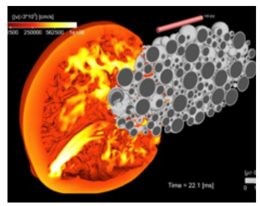
Trinity Platform Schedule Highlights 2015-2016

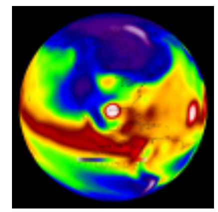
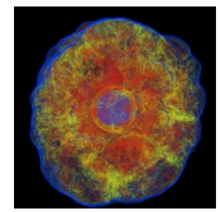
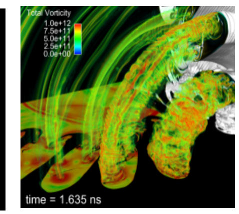


Trinity Update

- Trinity is the first instantiation of the ASC's ATS platform strategy
- SCC Physical Infrastructure Integration completed
- Trinity Testbeds delivered – Available June 2015
- Trinity Phase 1 delivered
- Trinity Center of Excellence activities ongoing at SNL, LLNL, and LANL
- Trinity System on schedule
- Next 6+ months
  – Trinity vendor integration, Aug-Oct 2015
  – Trinity Usage Model presentation to LANL, SNL, LLNL – August 2015
  – Acceptance Testing scheduled for December 2015
    - **Cray's Rhine/Redwood software is in schedule critical path**
  – Unclassified Science Runs for system stabilization, Jan-Feb, 2016
    - Rhine-Redwood system software
    - Burst Buffer Usage by applications

Trinity Platform Overview



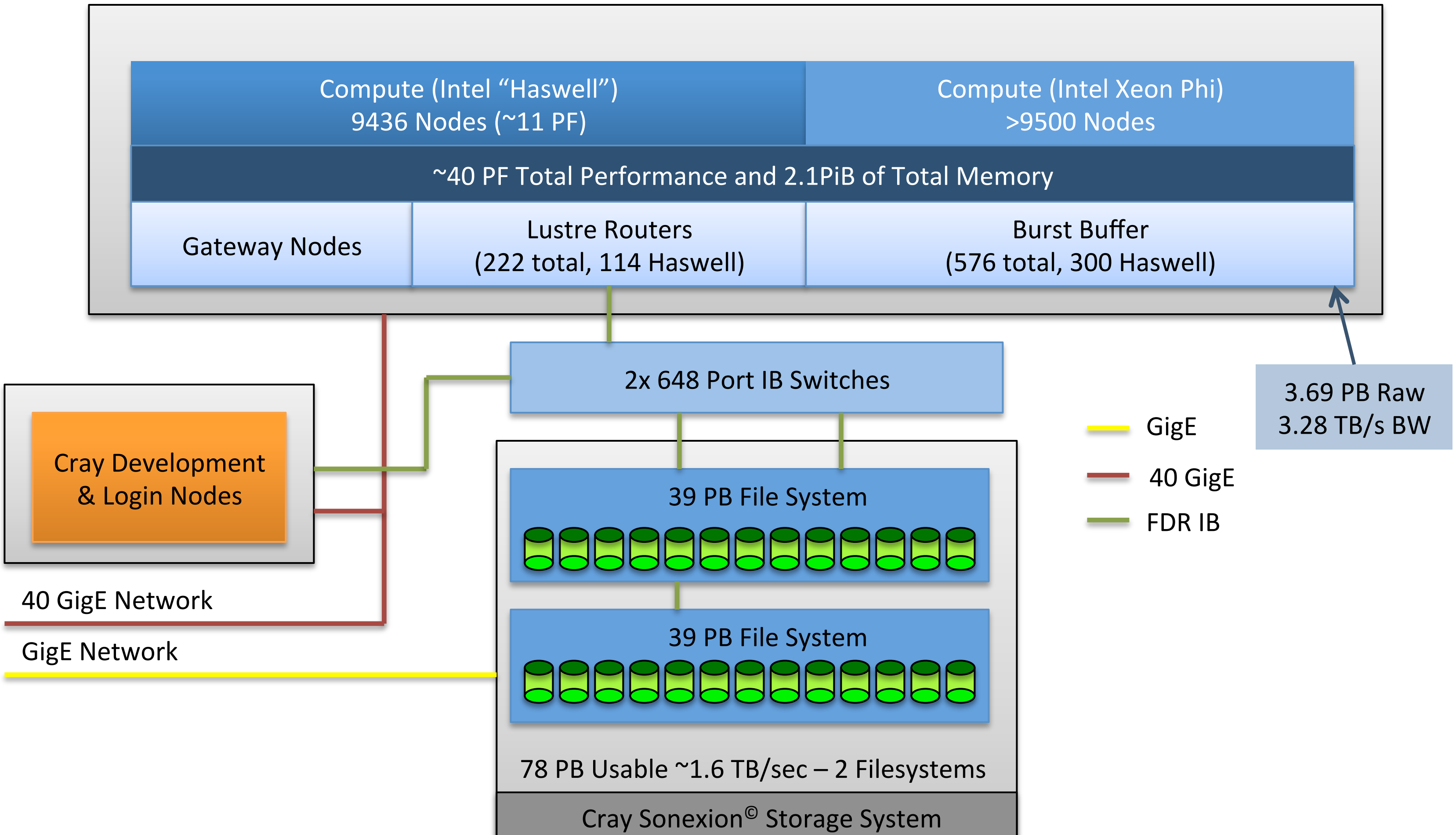




Scott Hemmert

August 2015

SAND2015-6938 PE

Trinity Architecture



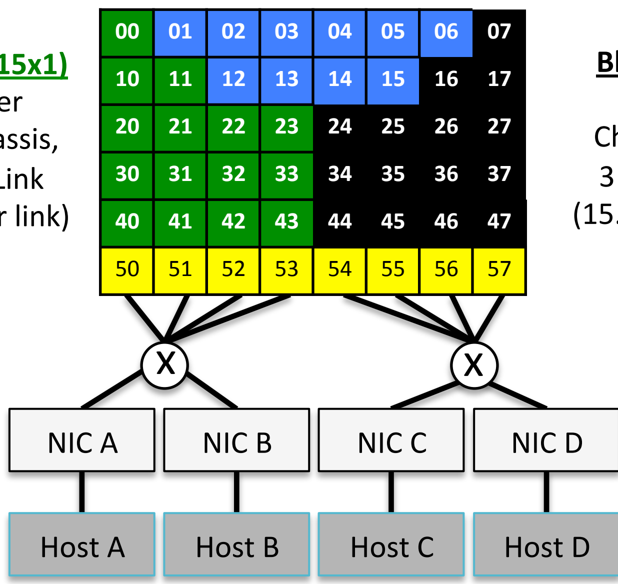
Cray Aries Interconnect

Cray Aries Blade

Blue Links (10x1)
To Other Groups,
10 Global Links
(4.7 GB/s per link)

Green Links (15x1)
To 15 Other
Blades in Chassis,
1 Tile Each Link
(5.25 GB/s per link)

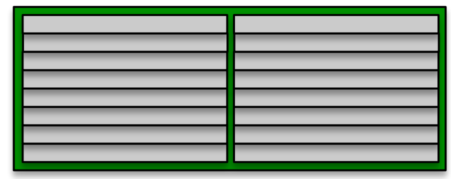
Black Links (5x3)
To 5 Other
Chassis in Group,
3 Tiles Each Link
(15.75 GB/s per link)



Gemini: 2 nodes, 62.9 GB/s routing bw
Aries 4 nodes, 204.5 GB/s routing bw

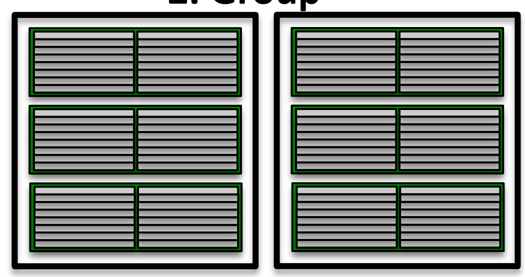
Aries has advanced adaptive routing

1. Chassis



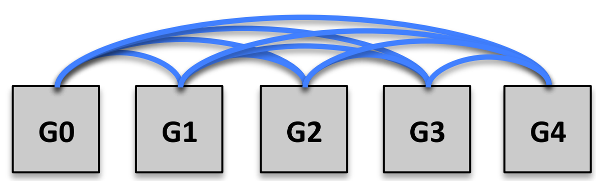
16 Blades Per Chassis
16 Aries, 64 Nodes
All-to-all Electrical Backplane

2. Group



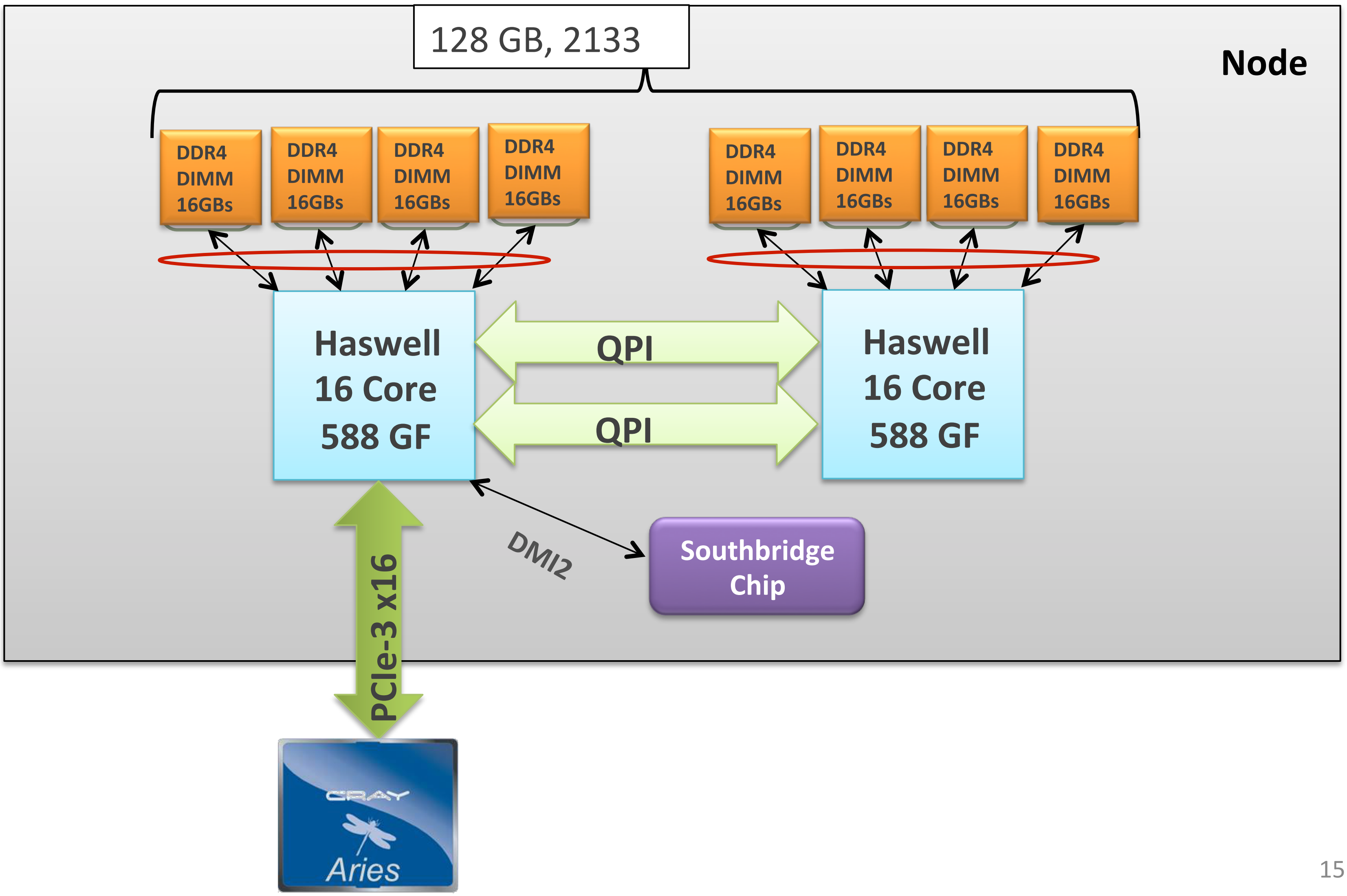
6 Chassis Per Group
96 Aries, 384 Nodes
Electrical Cables, 2-D All-to-All

3. Global

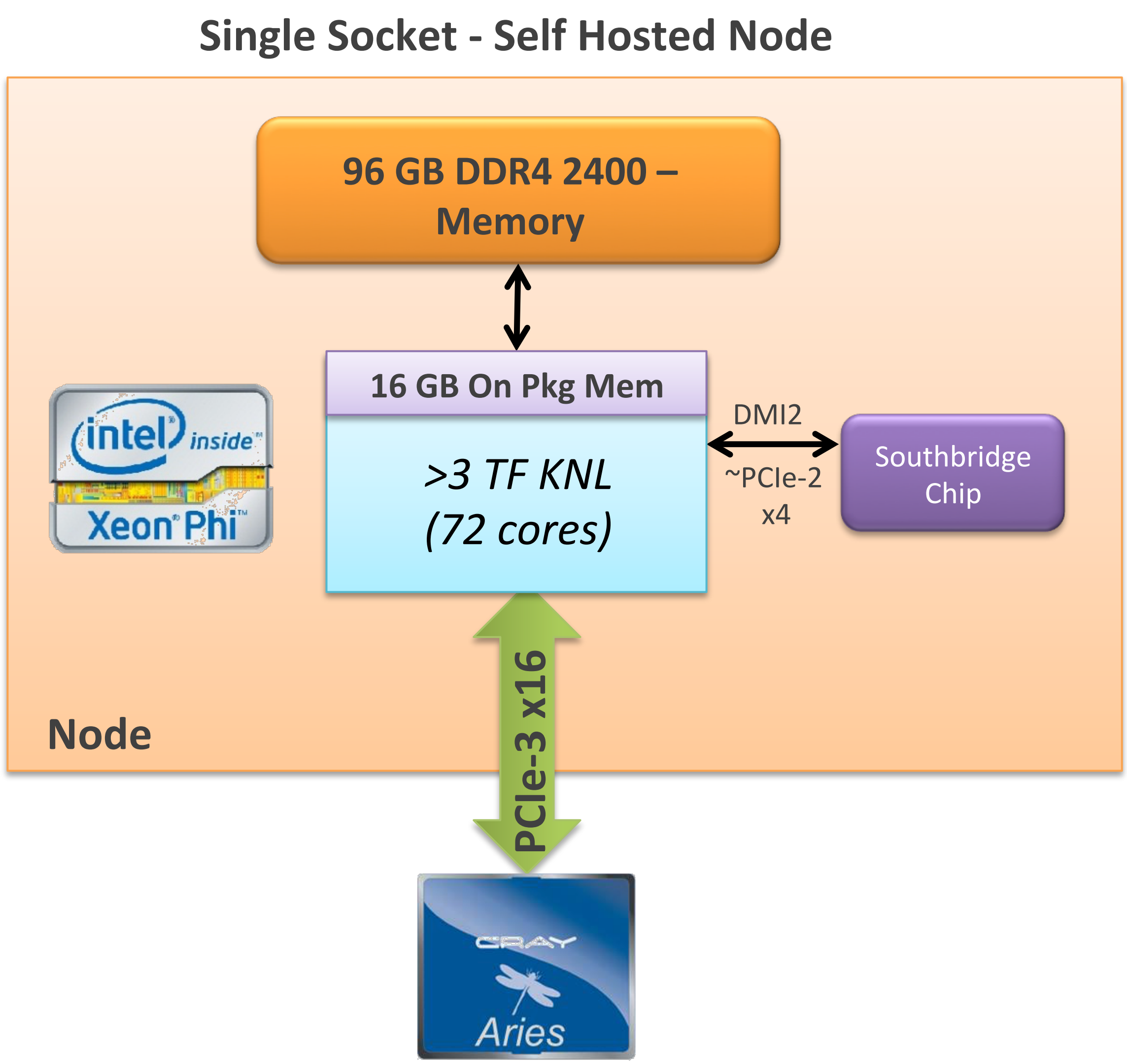


Up to 241 Groups
Up to 23136 Aries, 92544 Nodes
Optical Cables, All-to-All between Groups

Trinity Haswell Compute Node



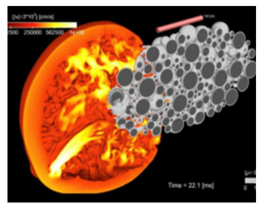
Trinity KNL Compute Node

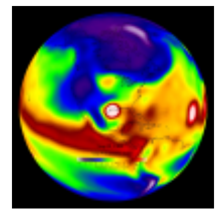
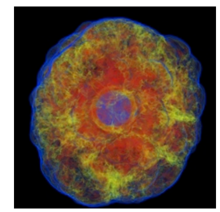
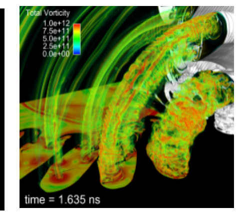


Test Bed Systems

- Gadget – Software Development Testbed
- Application Regression Testbeds
  - Configuration
    - 100 Haswell Compute Nodes
    - 720 TB / 15 GB/s Sonexion 2000 Filesystem
    - 6 Burst Buffer Nodes
  - Trinitite
    - LANL Yellow Network
  - Mutrino
    - Sandia SRN Network

Trinity File Systems Introduction and Usage Model



ACES File System Team

August 2015

LA-UR-15-24093

Thanks to all those who helped

- LANL: Kyle Lamb, Mike Mason, Terri Morris, & Alfred Torrez
- SNL: Ruth Klundt
- Cray: Chris Brady, Kim Schumann, & Mark Swan

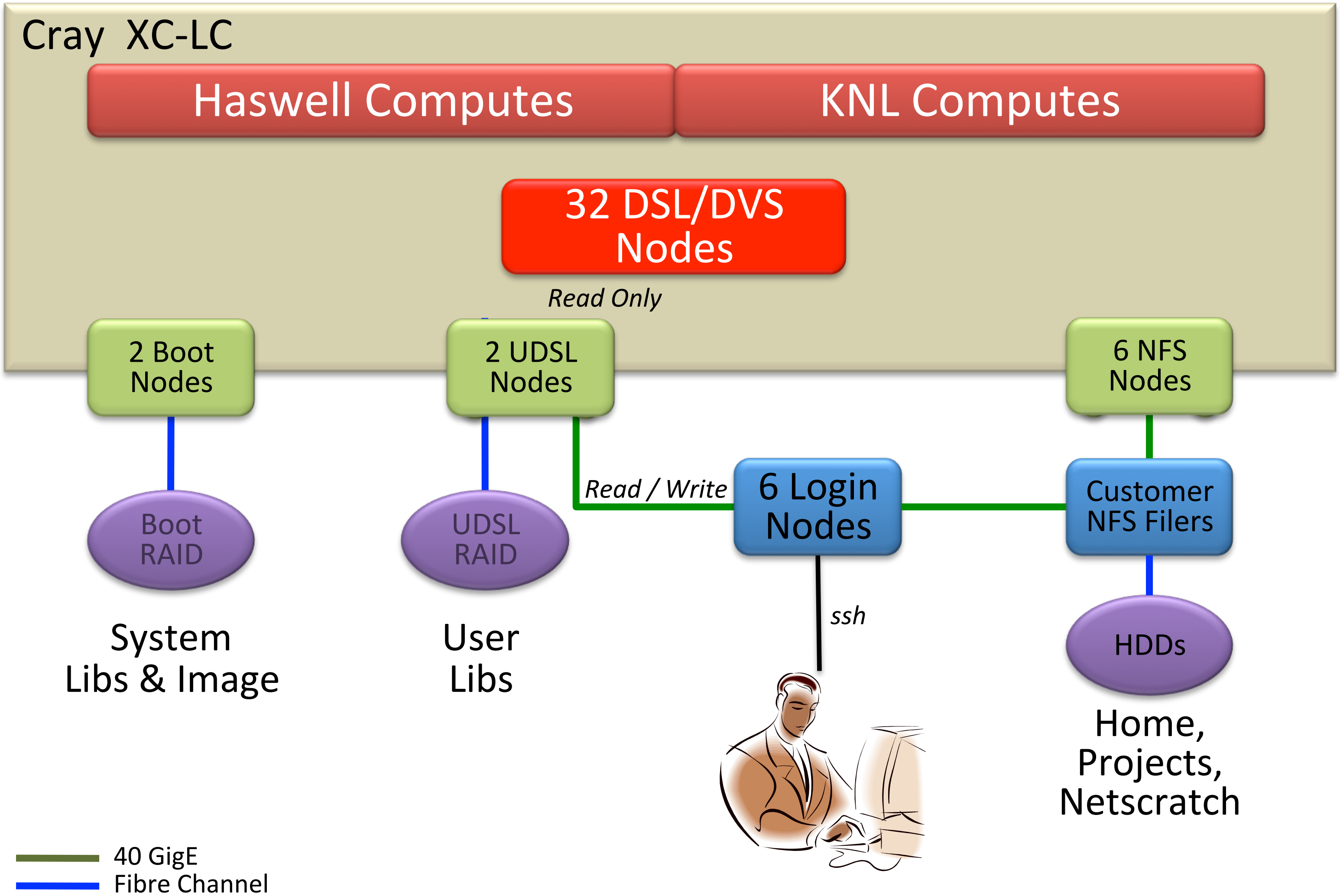
Trinity file systems & their purposes

- Home space
  - NFSv3 for user's own files
  - Use judiciously for writing application results
  - Backed-up and quotas
- Project space
  - NFSv3 to share such files as source code, common data sets, binaries, etc.
  - Backed-up and quotas
- Packages space
  - NFSv3 to share tools and applications, especially 3rd Party, that are useful to many code teams but made available by one person
  - Backed-up and quotas
- Netscratch space
  - NFSv3 for faster serial access to restart, graphics, time-step data, etc.
  - Not backed-up and quotas
- Scratch space
  - Lustre for fast parallel access to restart, graphics, time-step data, etc.
  - Not backed-up, but purged
- UDSL (User Dynamic Shared Library) space
  - NFSv3 for user/code-specific libraries read by applications at launch
  - Read-only from computes; write from login nodes
- System library and image space
  - NFSv3 for OS images and OS/installed product libraries
  - Cfengine-managed

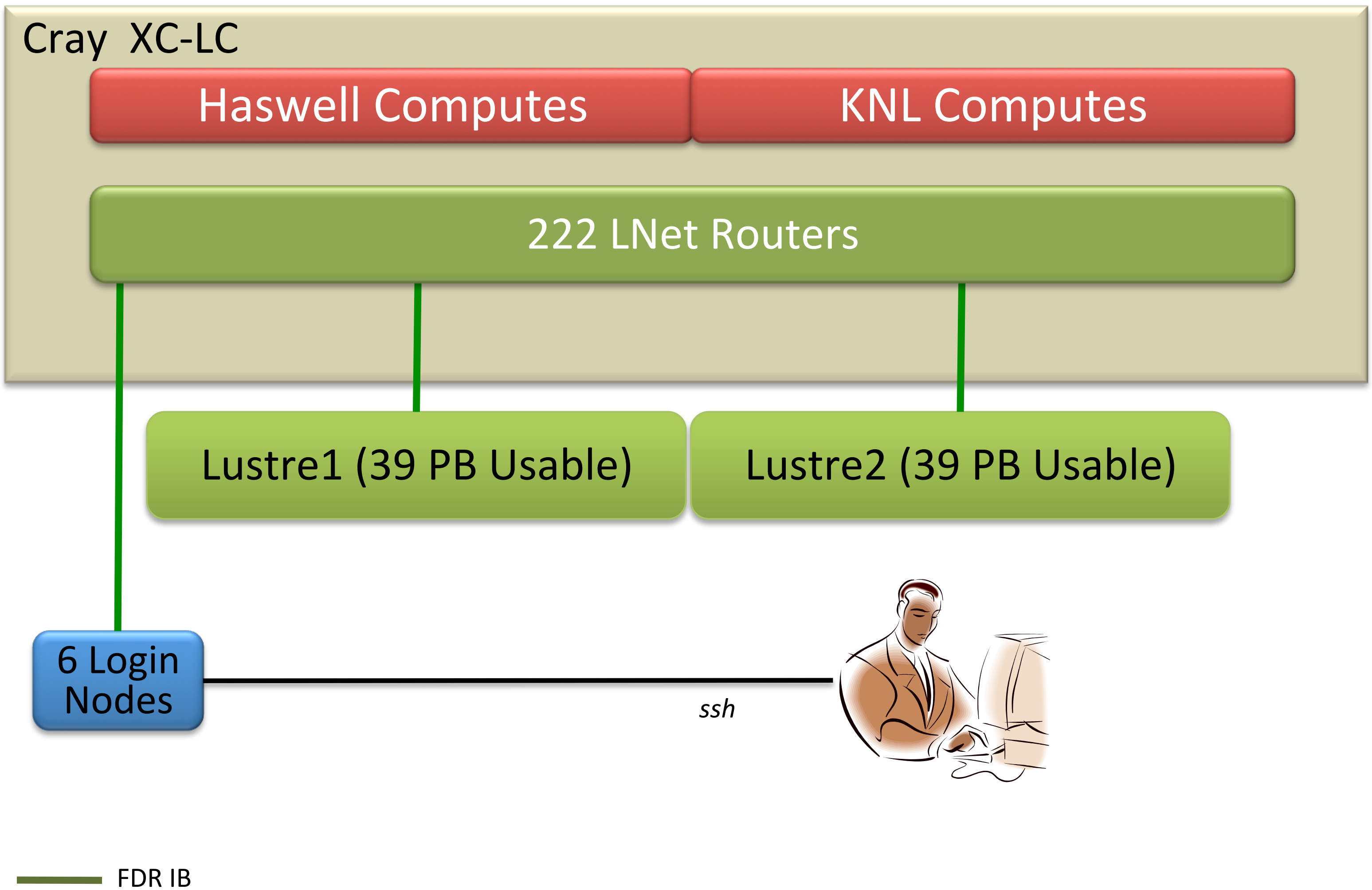
Things to know about Trinity's Lustre PFSes

- Capacities (usable)
  - Trinity: 39 PB per PFS
  - Trinitite & Mutrino: 720 TB
  - Gadget: 360 TB
- Sustained bandwidth (7.5 GB/s/SSU)
  - Trinity: 810 GB/s per PFS (108 SSUs/PFS)
  - Trinitite & Mutrino: 15 GB/s (2 SSUs)
  - Gadget: 7.5 GB/s (1 SSU)
- OST counts (2 per SSU)
  - Trinity: 216 per PFS
  - Trinitite & Mutrino: 4
  - Gadget: 2
- Use GridRAID

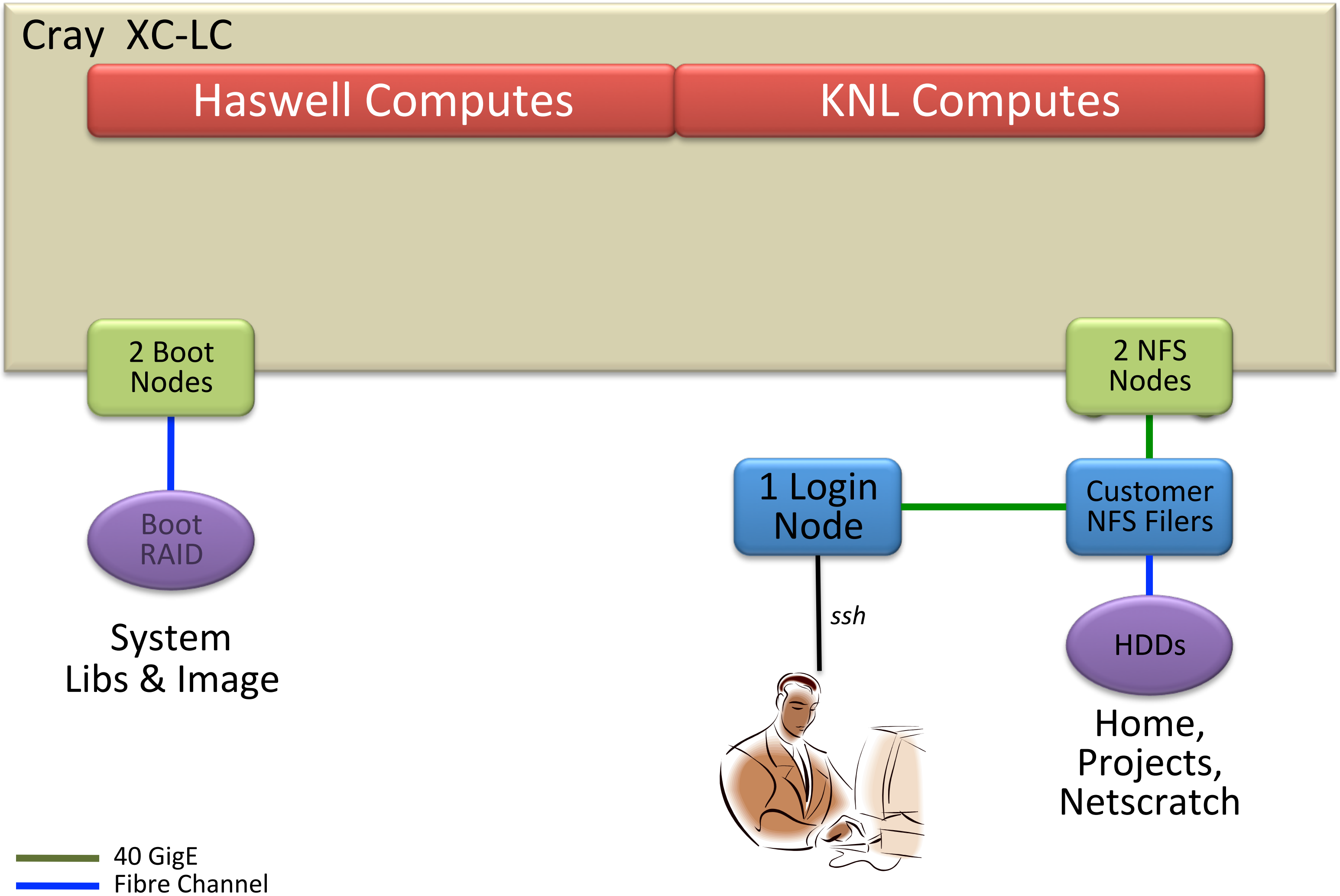
Trinity file systems



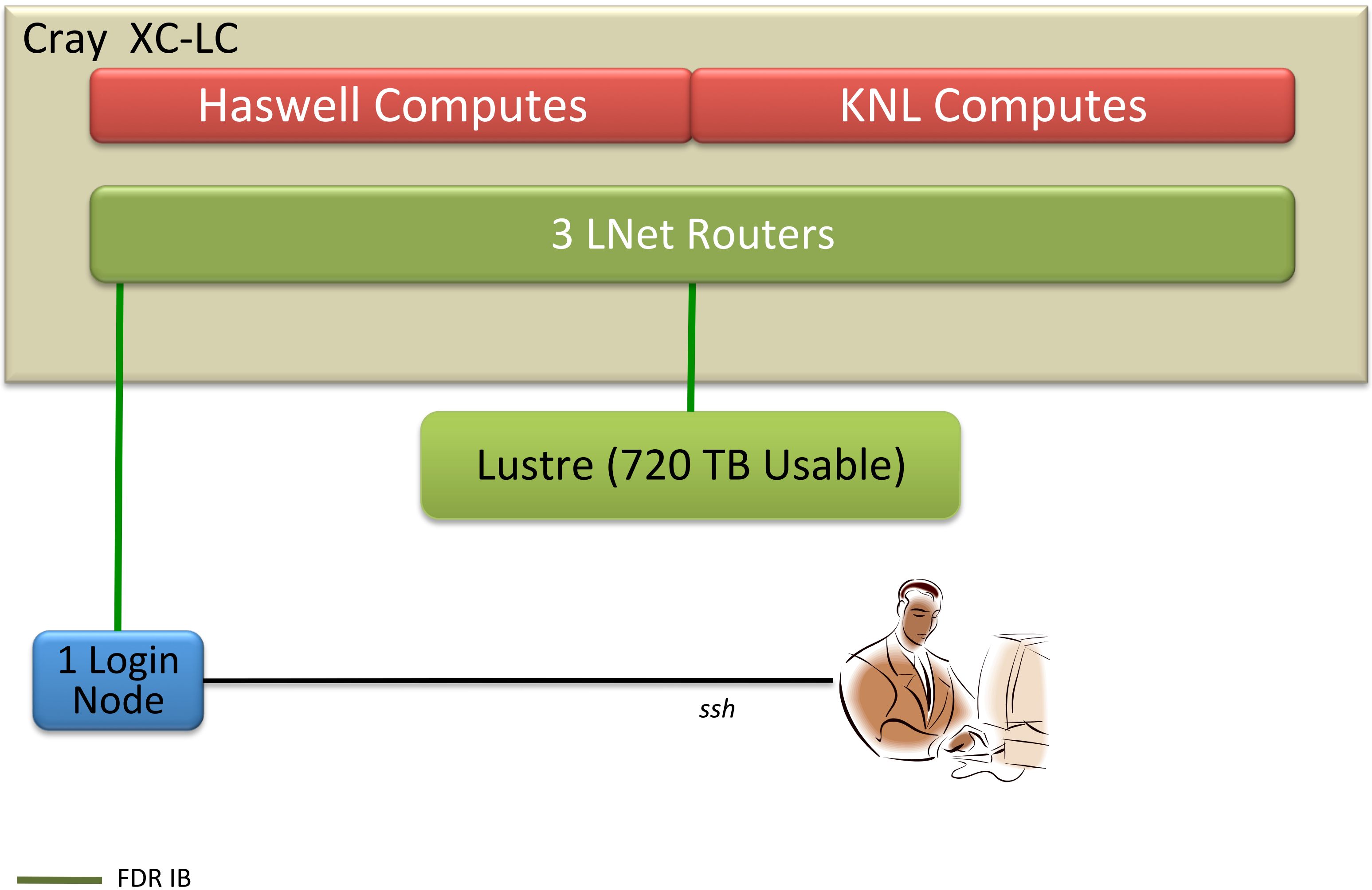
Trinity parallel file systems



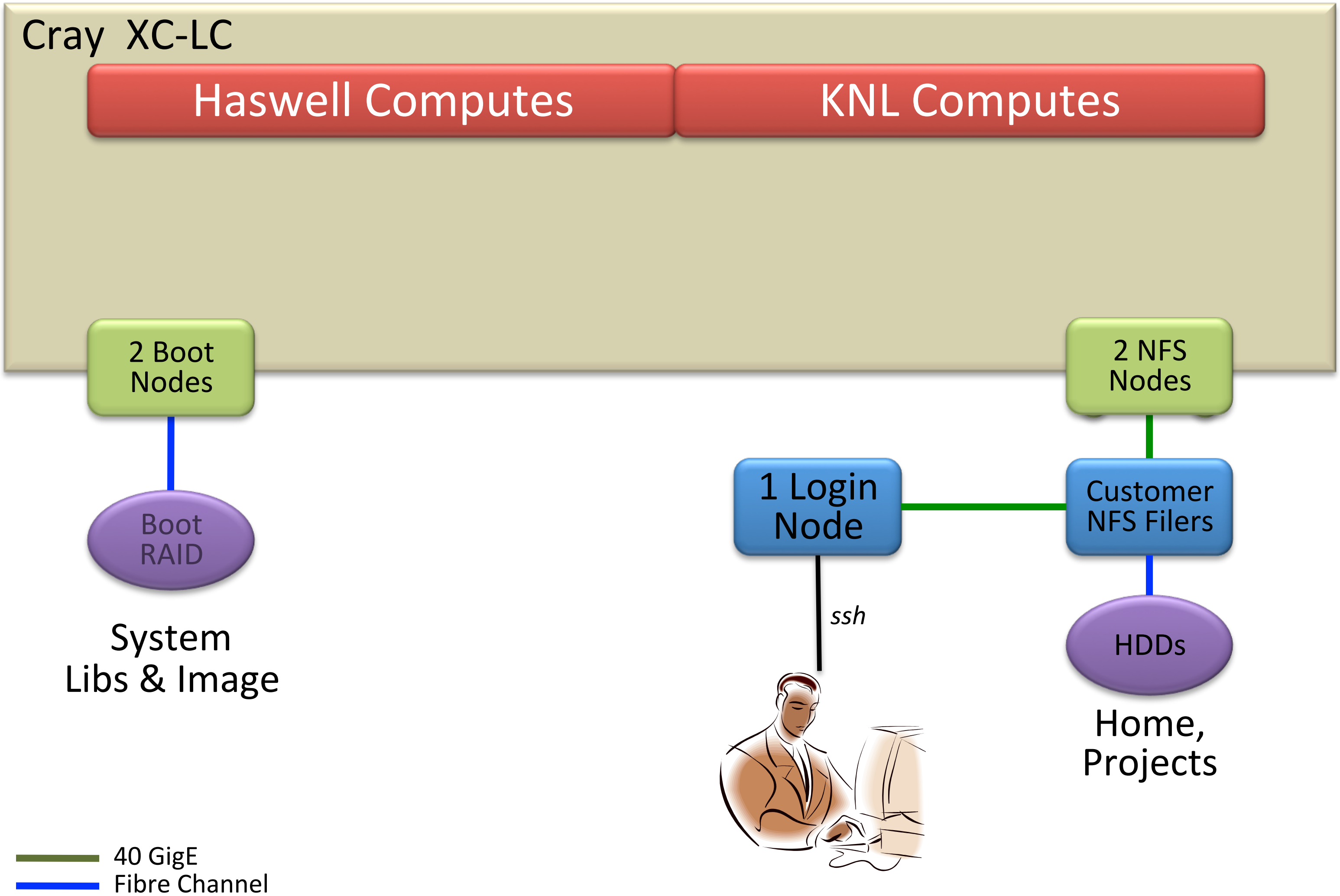
Trinitite file systems



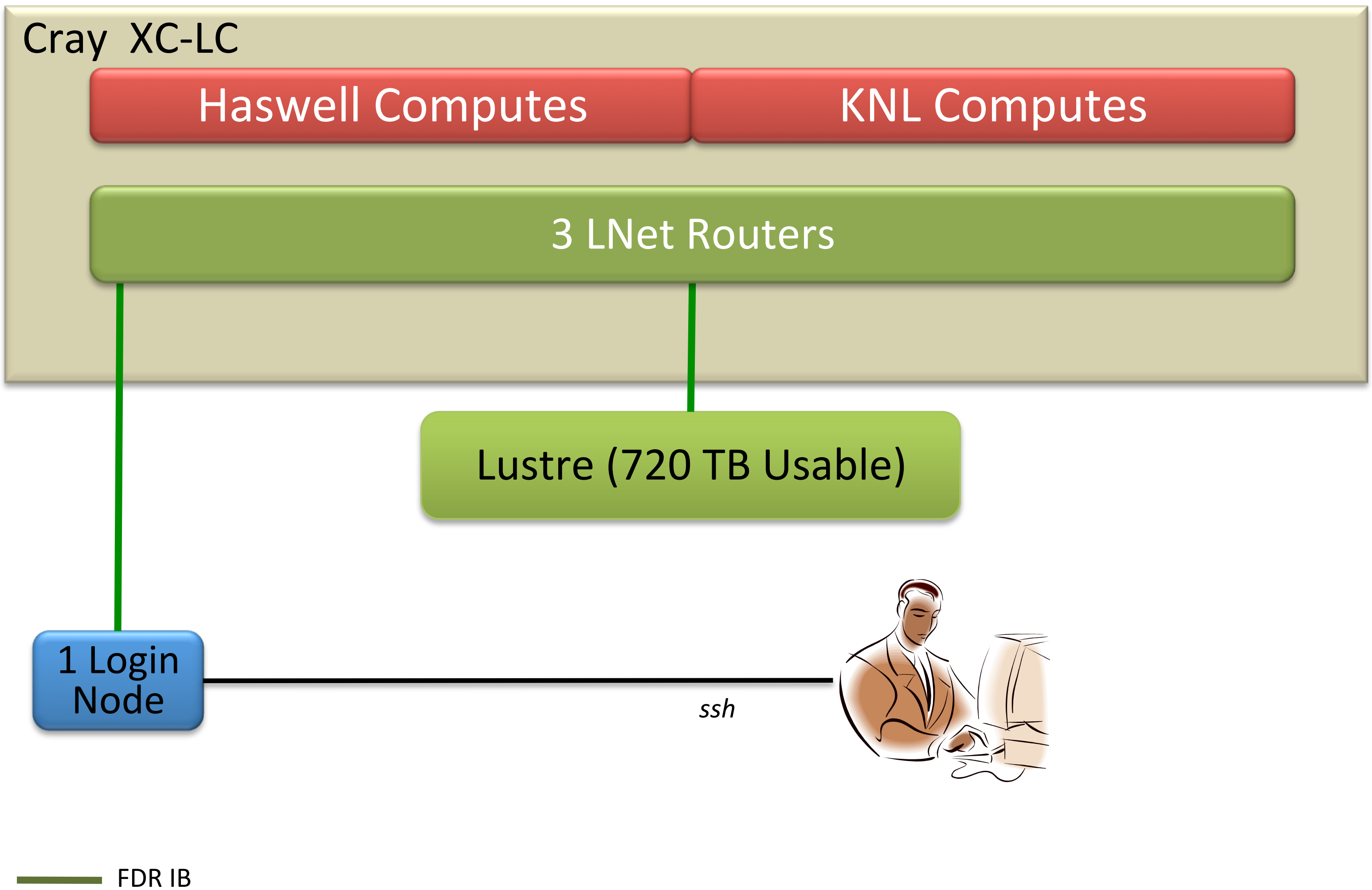
Trinitite parallel file systems



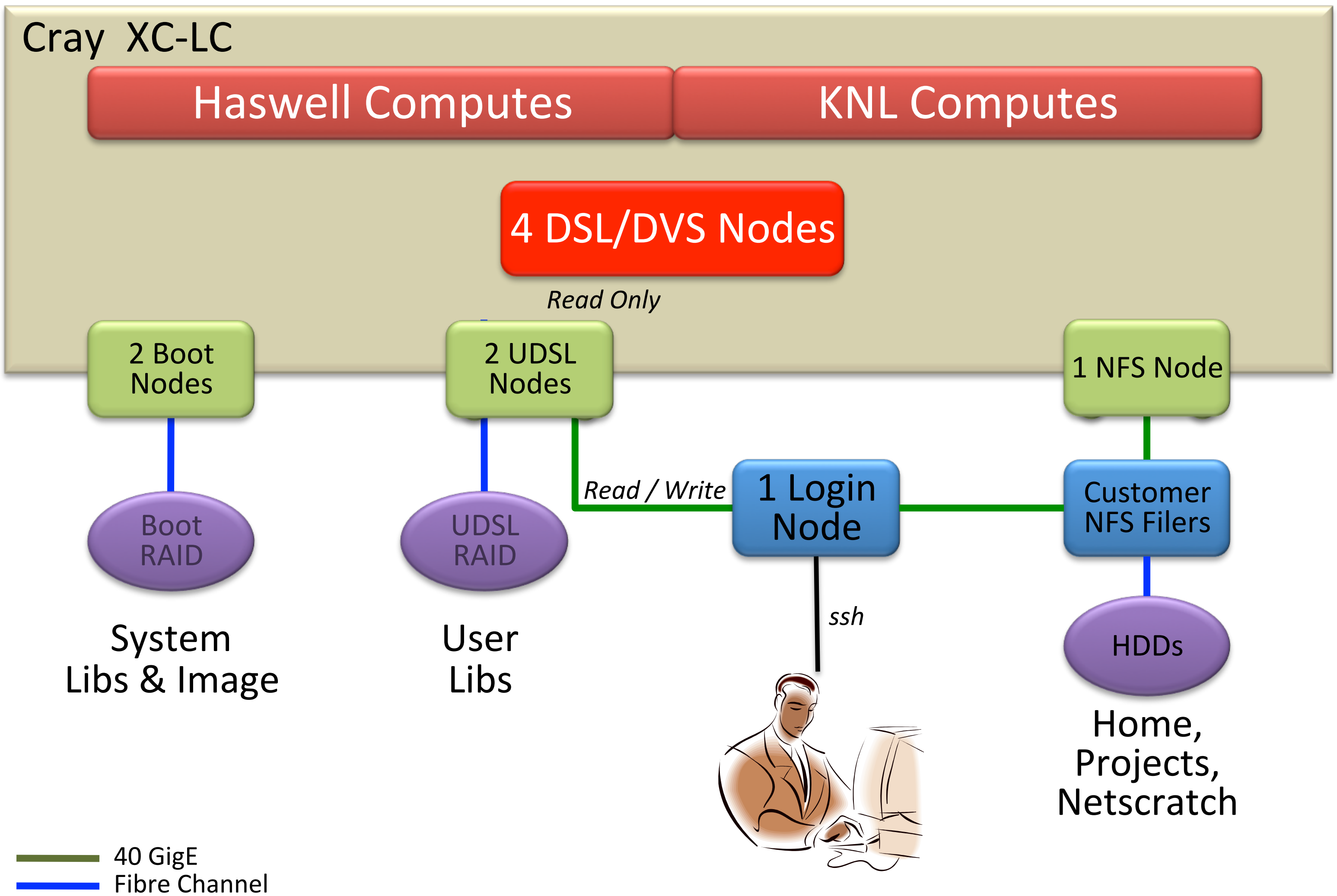
Mutrino file systems



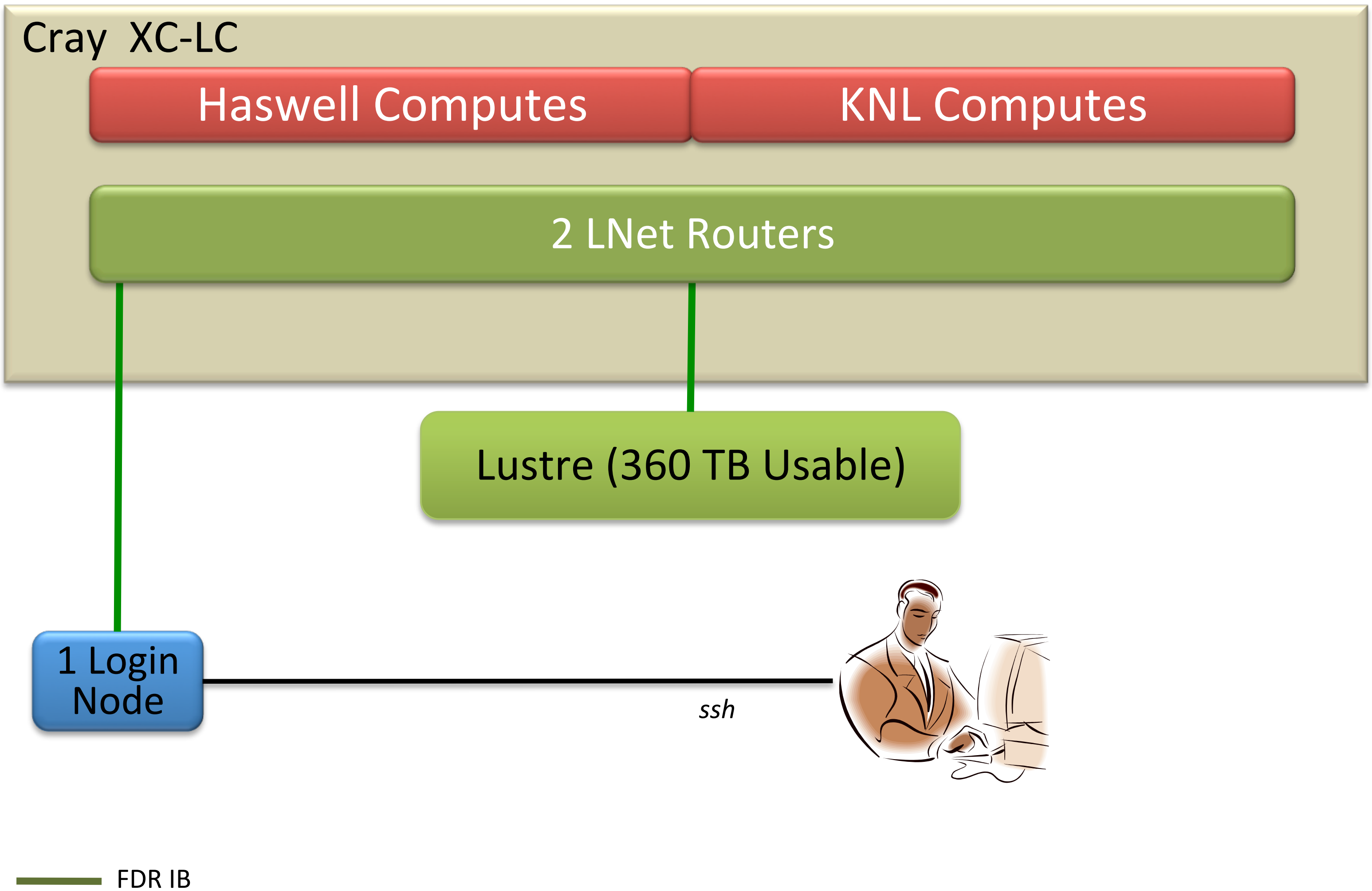
Mutrino parallel file systems



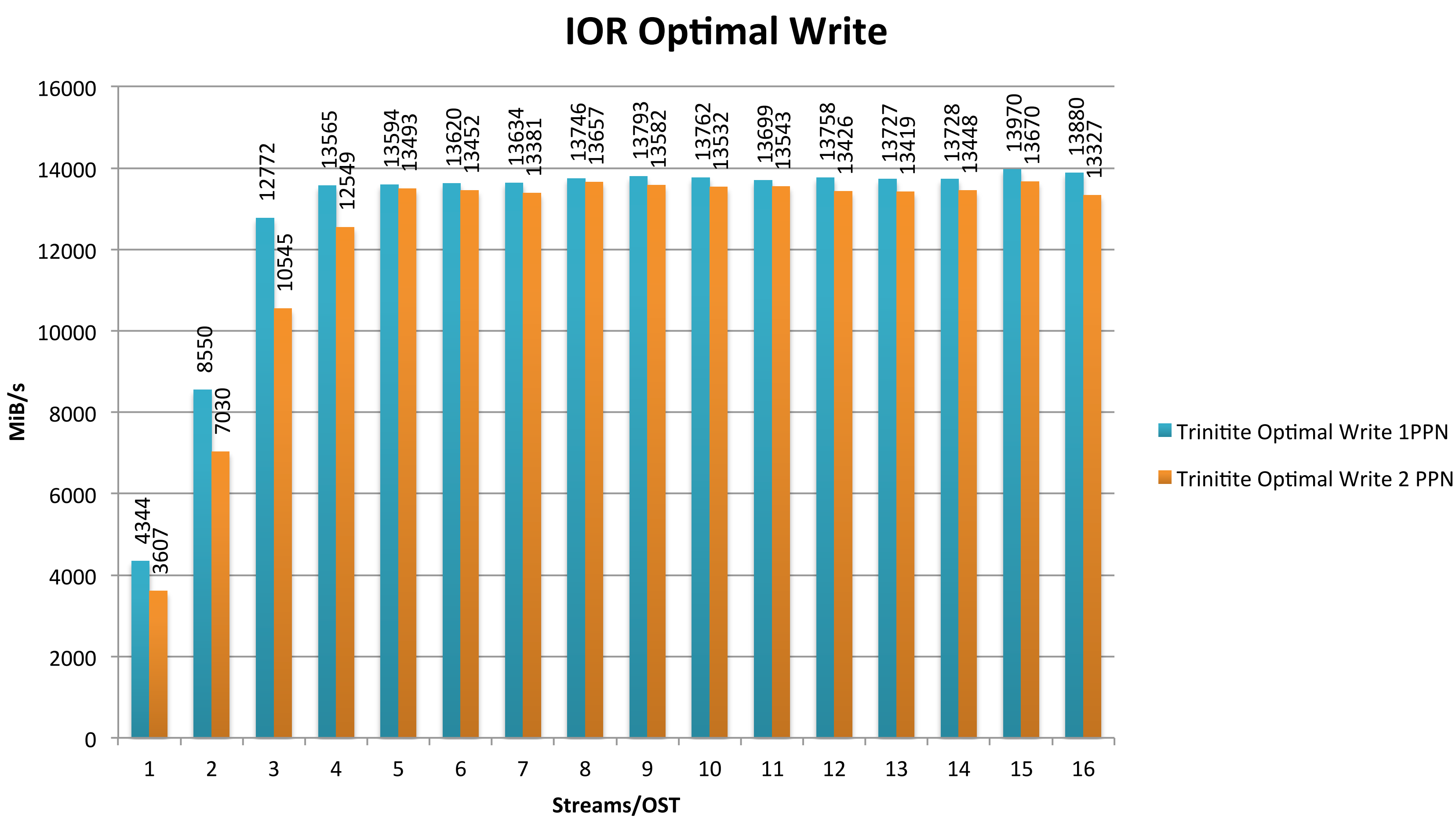
Gadget file systems



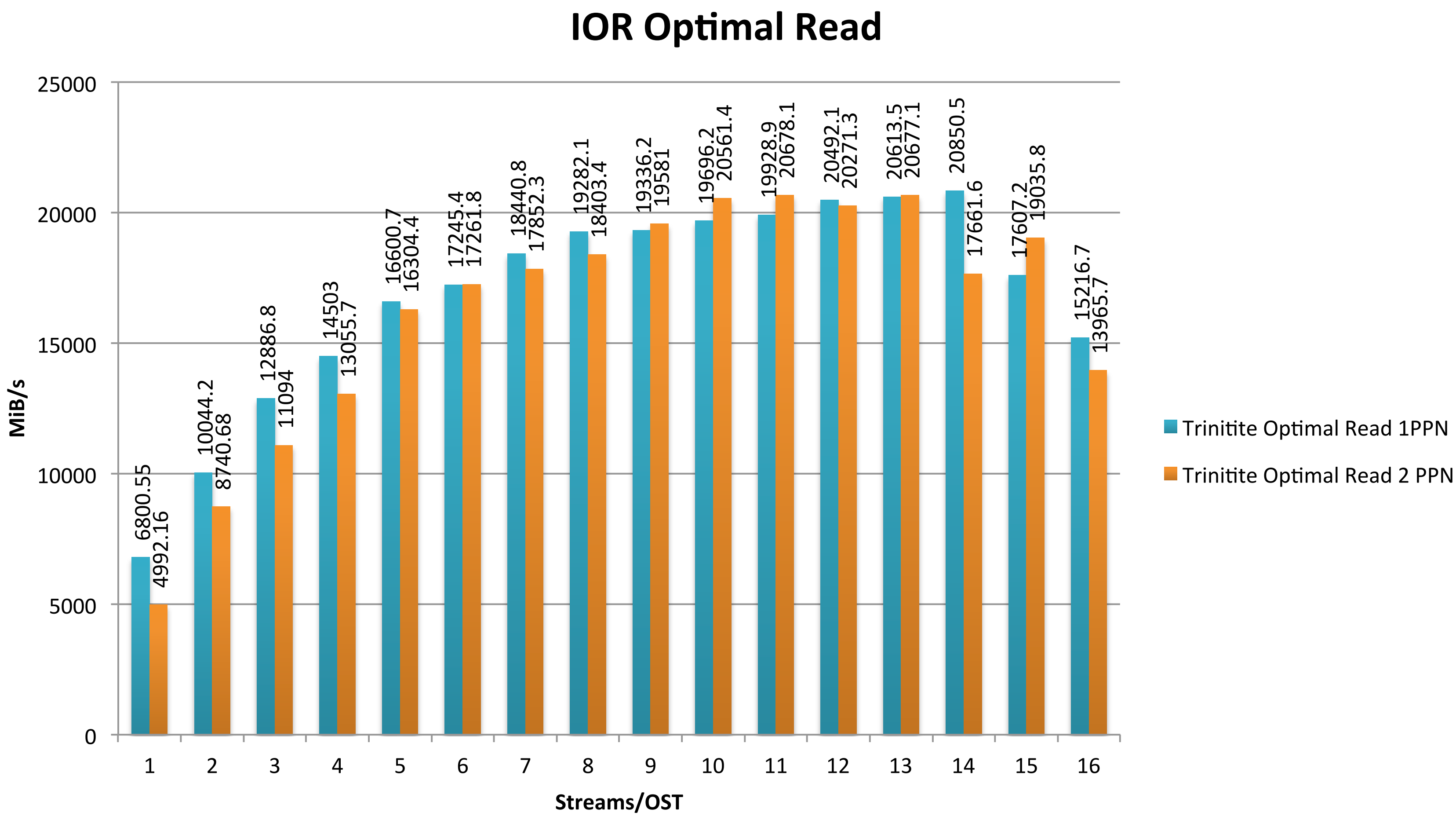
Gadget parallel file systems



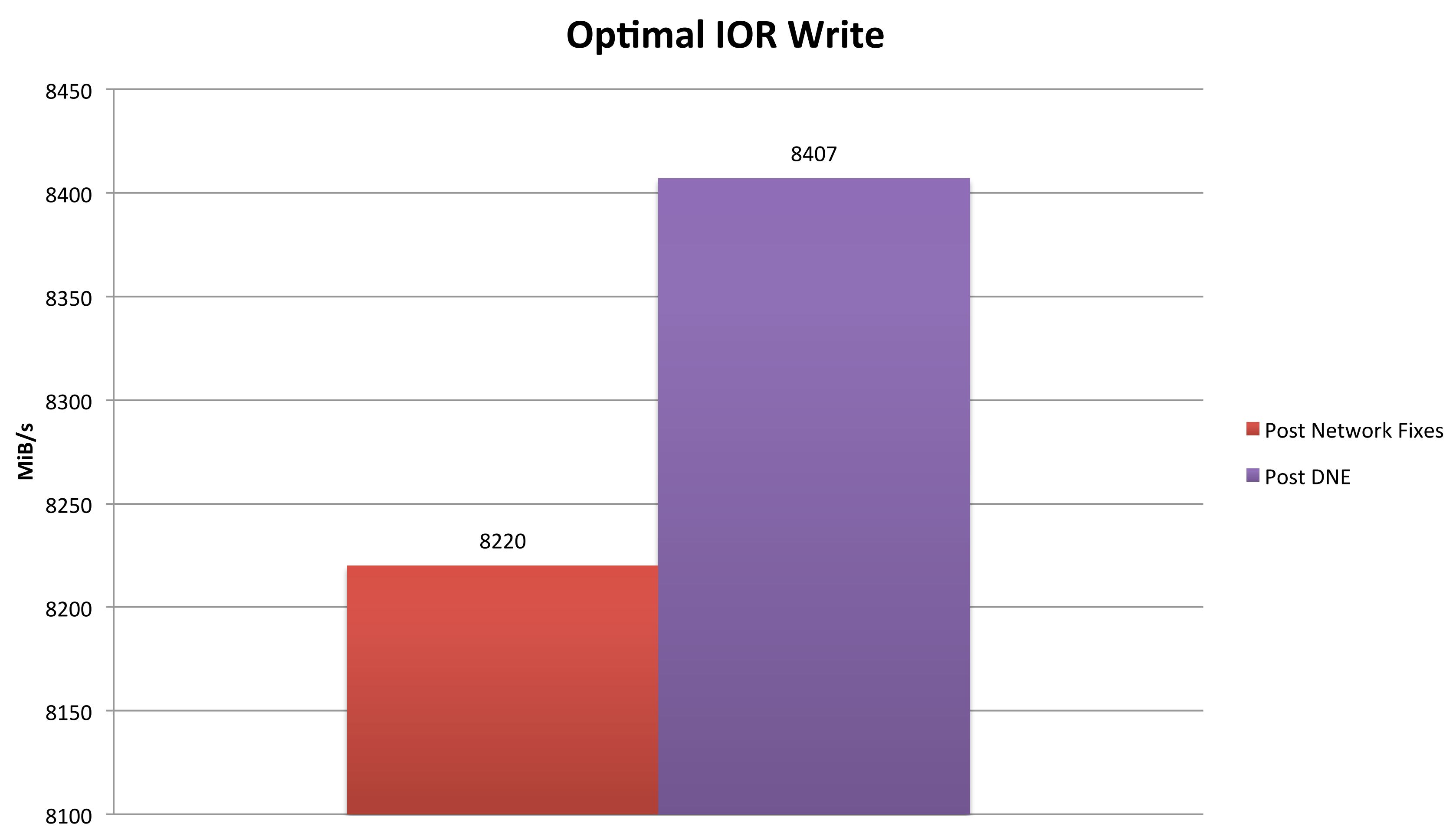
Measurements suggest 5-16 I/O streams/OST for optimal write performance



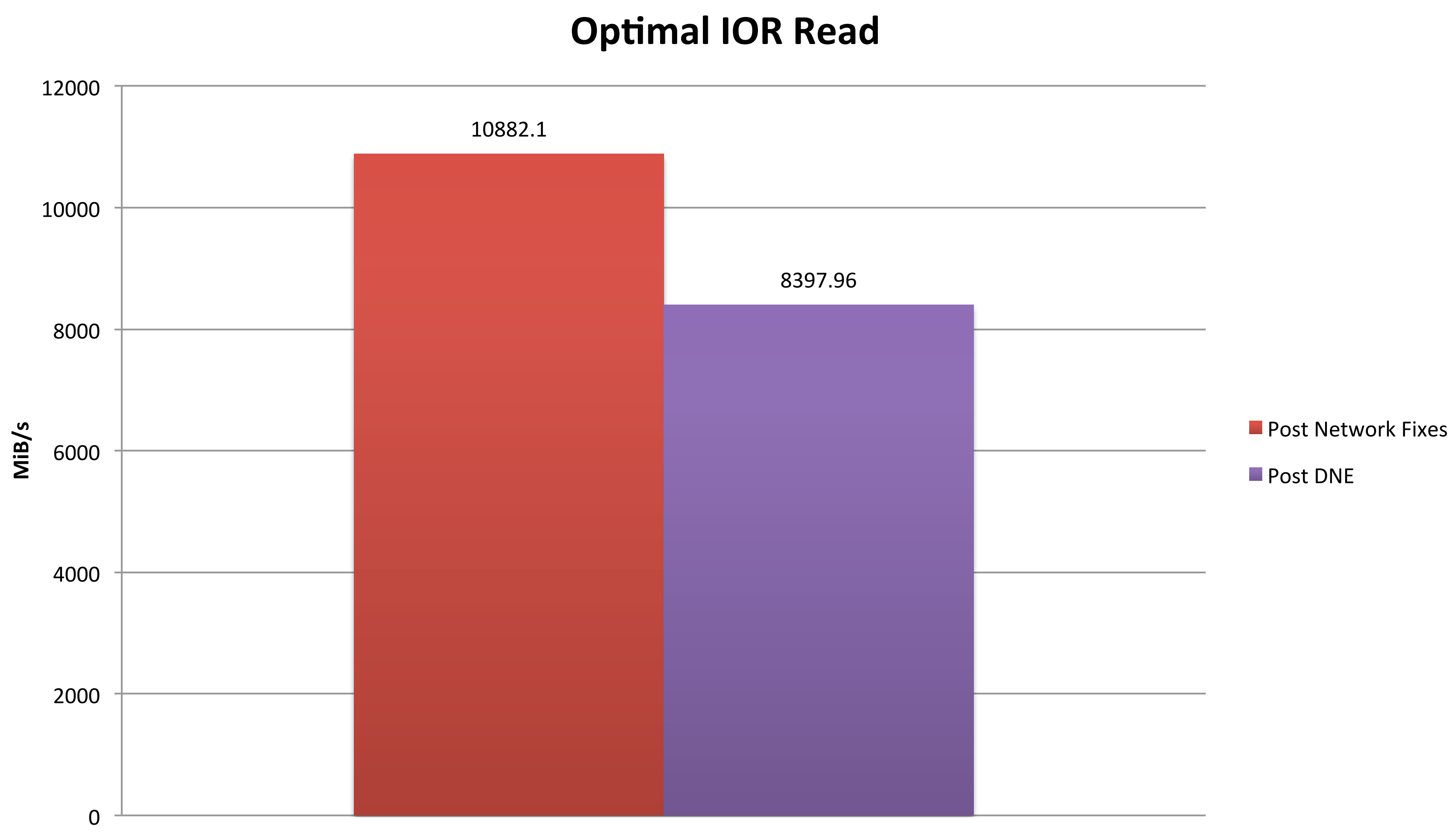
Measurements suggest 10-14 I/O streams/ OST for optimal read performance



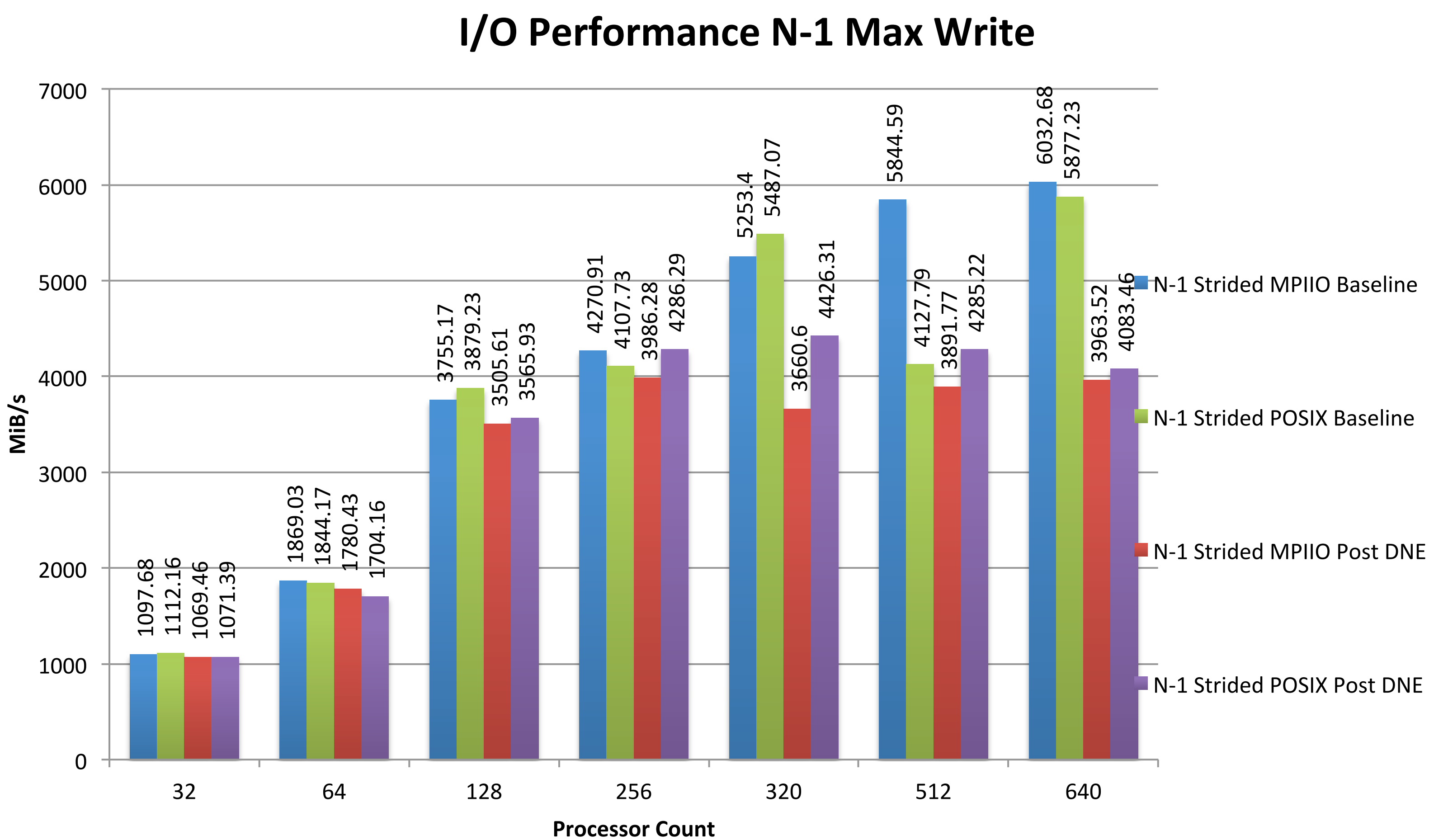
Gadget optimal N-N IOR write



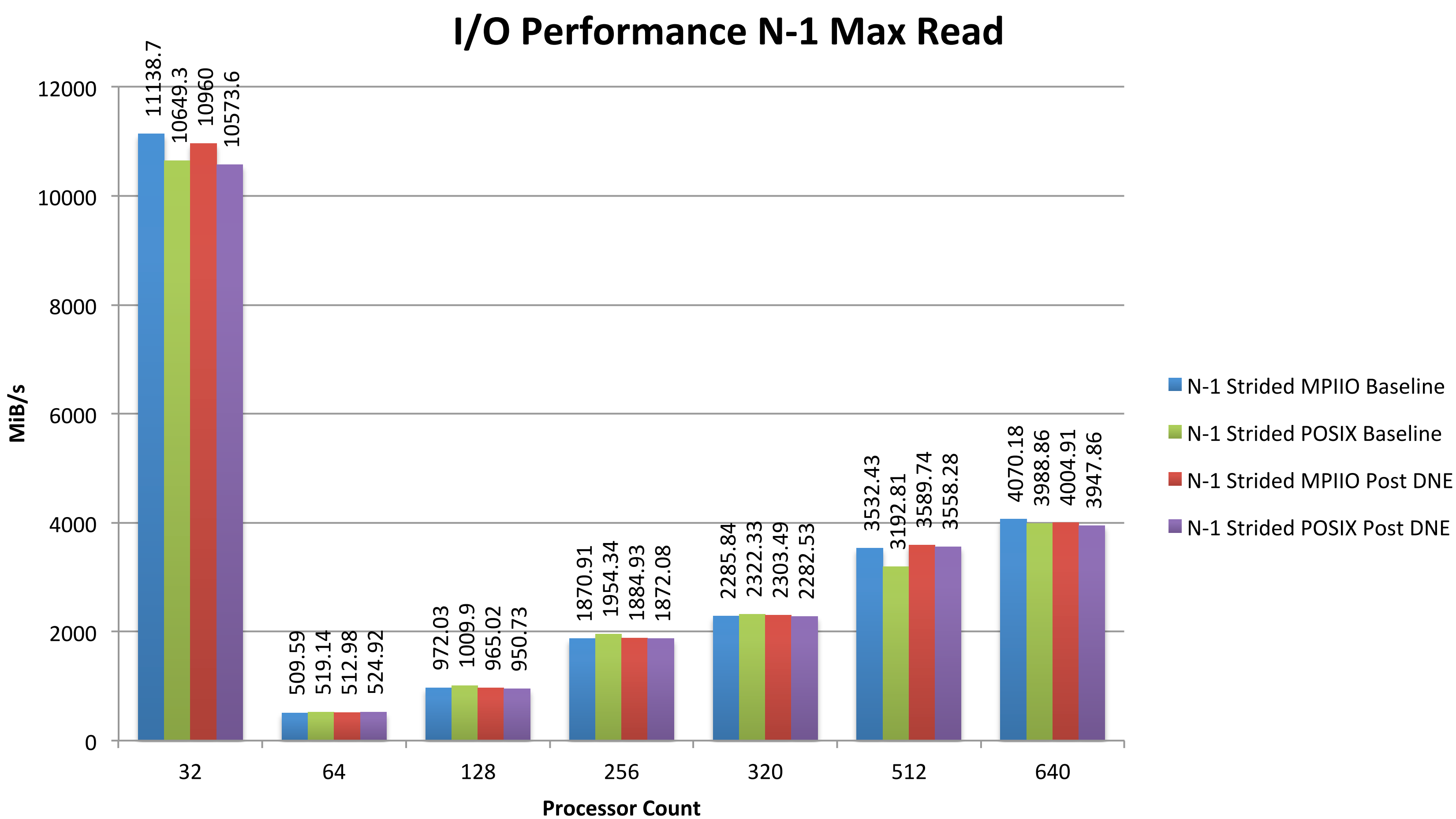
Gadget optimal N-N IOR read



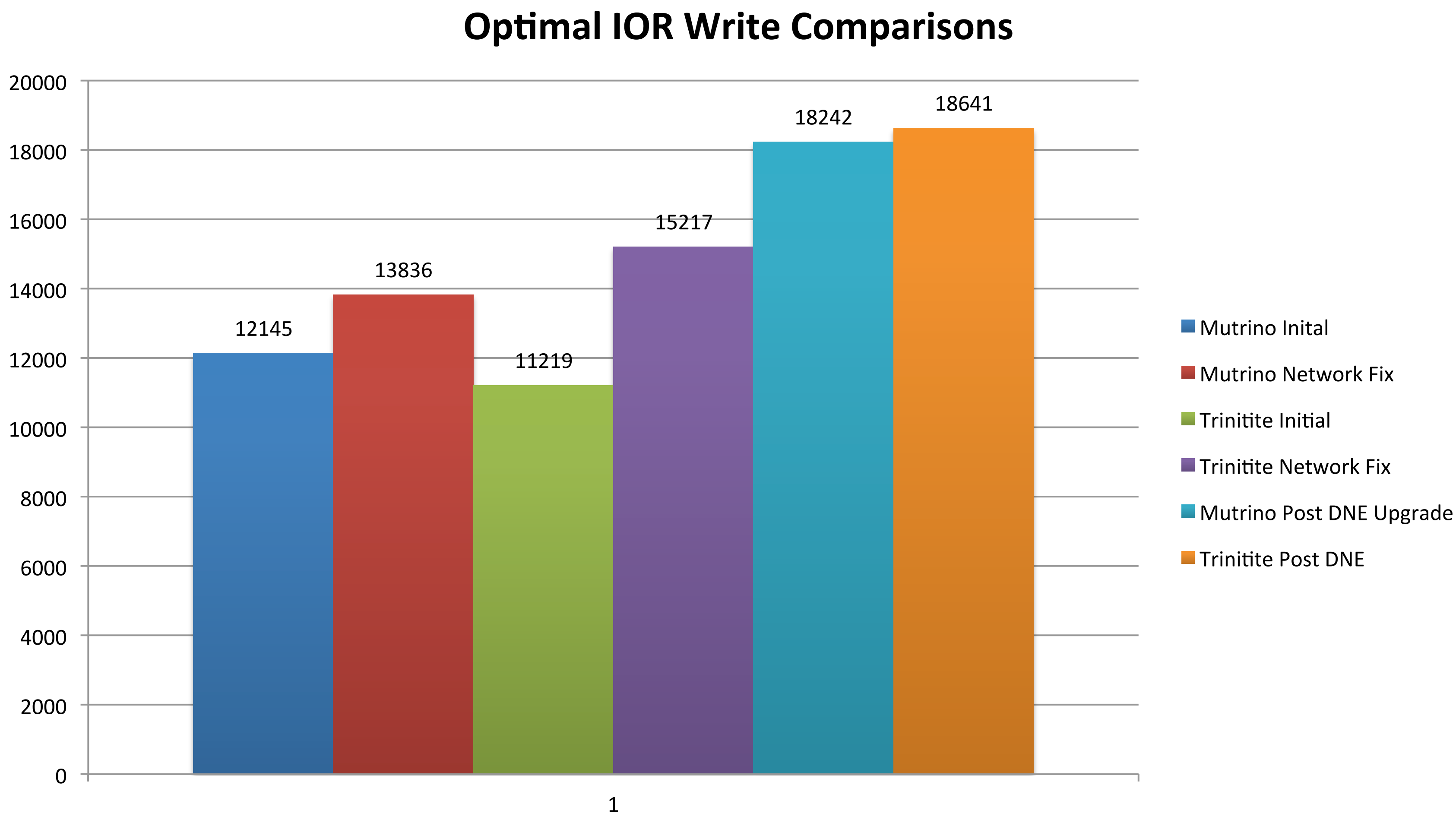
Gadget N-1 strided IOR write



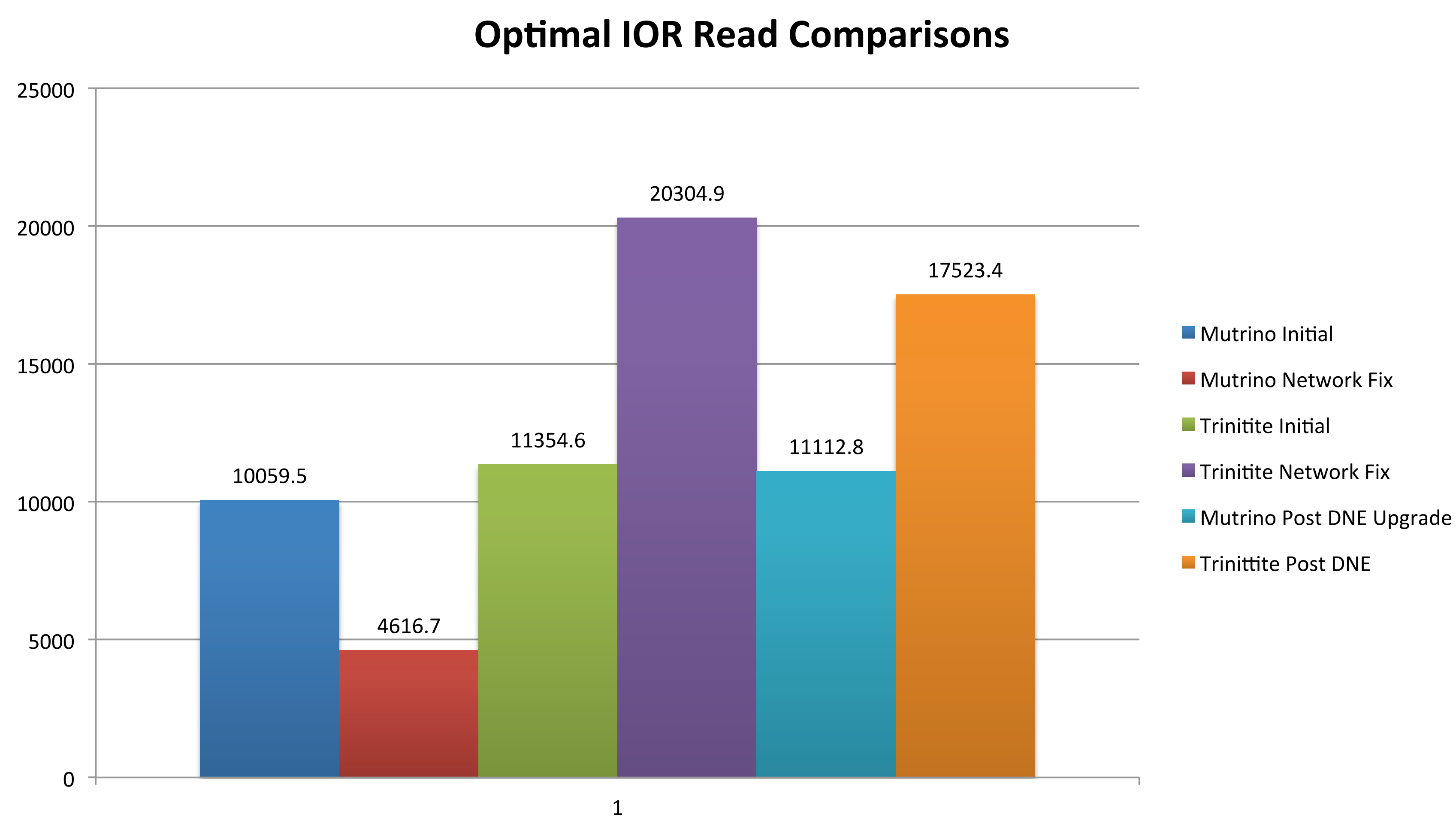
Gadget N-1 strided IOR read



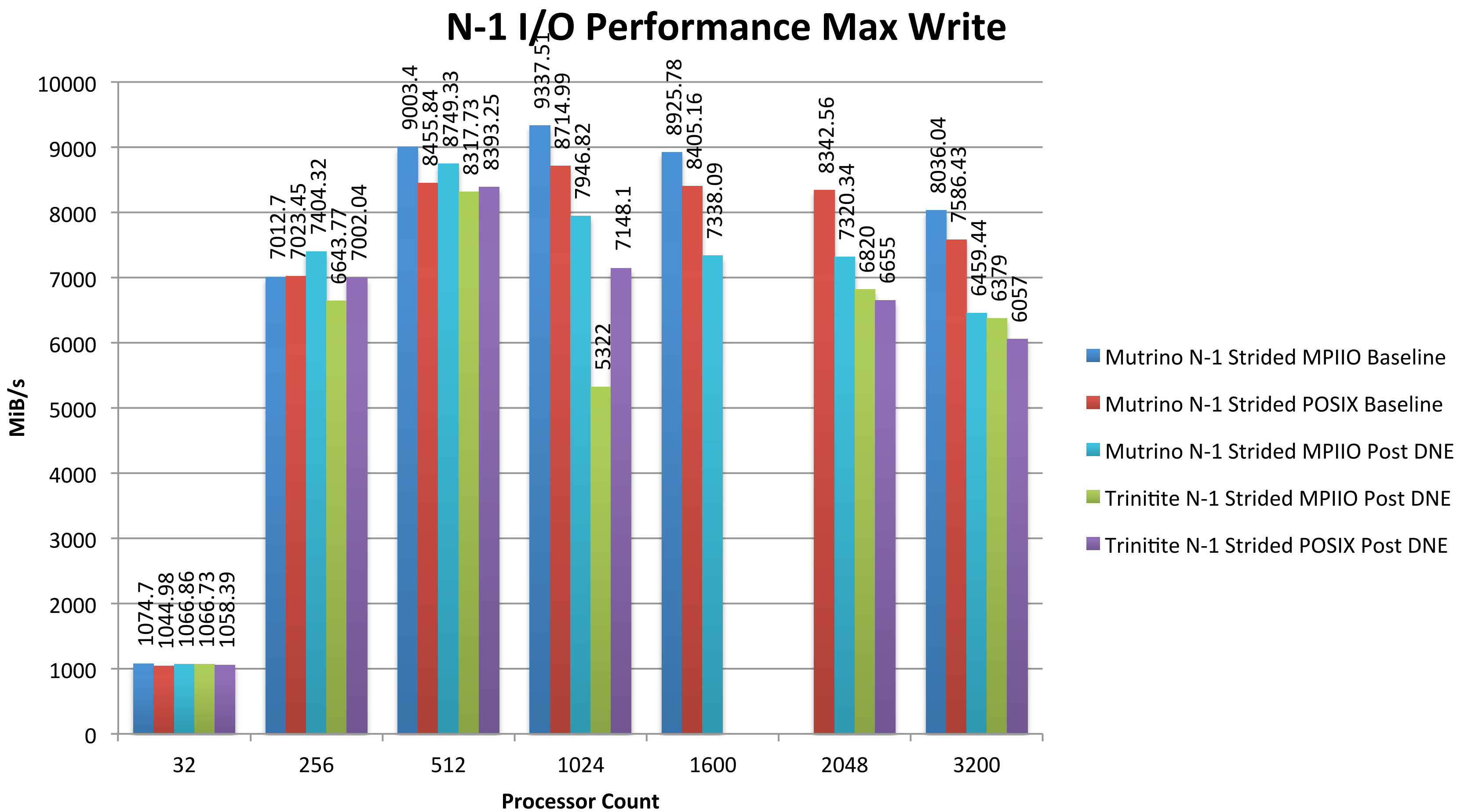
Trinitite & Mutrino optimal N-N IOR write



Trinitite & Mutrino optimal N-N IOR read

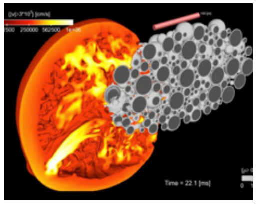
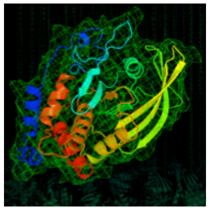
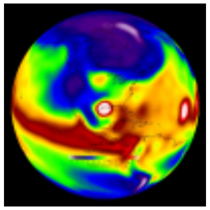
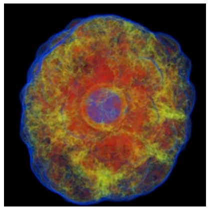
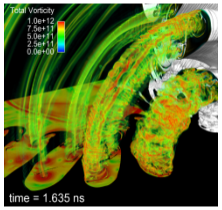


Trinitite & Mutrino N-1 strided IOR write



Trinitite & Mutrino N-1 strided IOR read

Trinity Usage Model

Karen Haskell
Jeff Johnson
August 2015

LA-UR 15-xxxxx

ASC Usage Models

- The Trinity Usage Model is our "contract" with users for how to best use the machine.
- The support model, form and functionality described should be familiar to users of Cielo.
- The Usage Model will form baseline requirements for the FY17 ASC L2 ATCC Production Readiness Milestone.
- Like Cielo, Trinity will be jointly managed by LANL and SNL; management by two labs rather than a single laboratory.
- As an ACES tri-lab ASC platform, policies and operations may differ from those of local laboratory platforms.

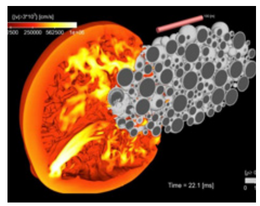
Availability

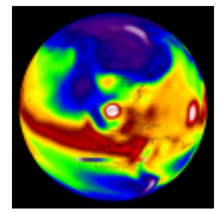
- Final phase 1 document will be available by ATCC-1
- Not intended to be a living document, though it will be updated for Trinity phase 2
- Widely distributable, documenting the Trinity system at the time of publication
- Intended as an agreement with the users of the system as to how the machine will be configured and managed

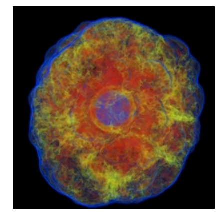
Feedback

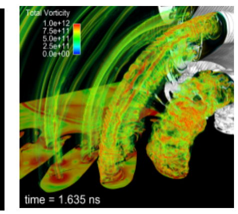
- We will incorporate feedback from this series of meetings
  - Email tum-editors@lanl.gov with questions or comments
- Today's presentations and the Usage Model will be available at http://trinity.lanl.gov/ and on the ACES web pages at http://aces.sandia.gov

Trinity Platforms User Support









ACES User Support Team

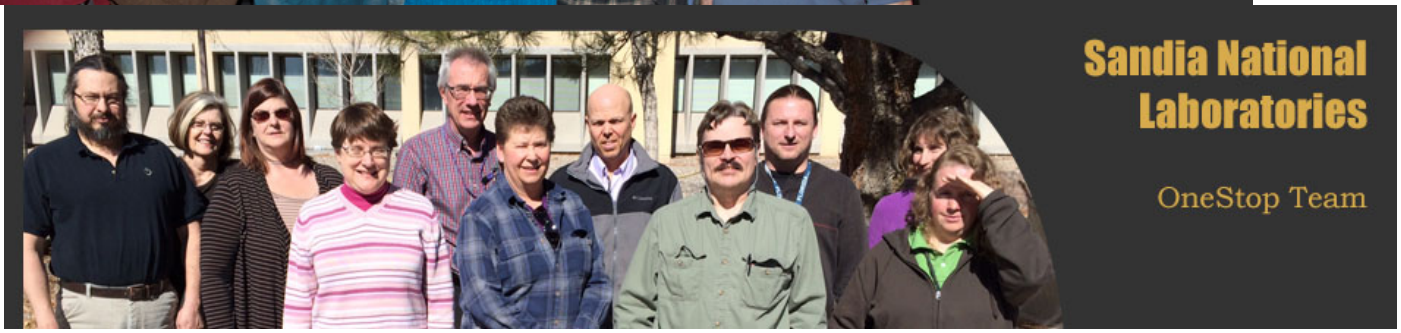
August 2015

LA-UR 15-xxxxx

ACES User Support

Similar Contact Information to Cielo

- Help available 8am to 5pm Mountain Time M-F
- Phone: 1-877-378-1110
- email: trinity-help@sandia.gov



ACES User Support

Documentation

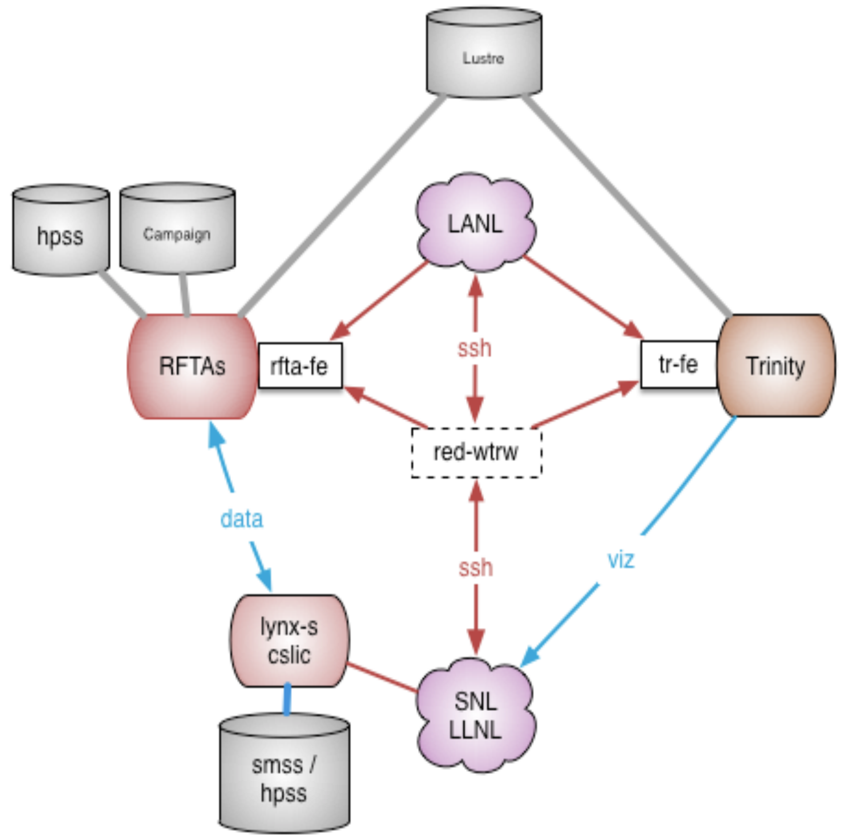
- https://aces.sandia.gov
- Web pages also accessible on Trinity filesystems: `/usr/projects/hpcdoc/hpc.lanl.gov`

Allocations and Accounts

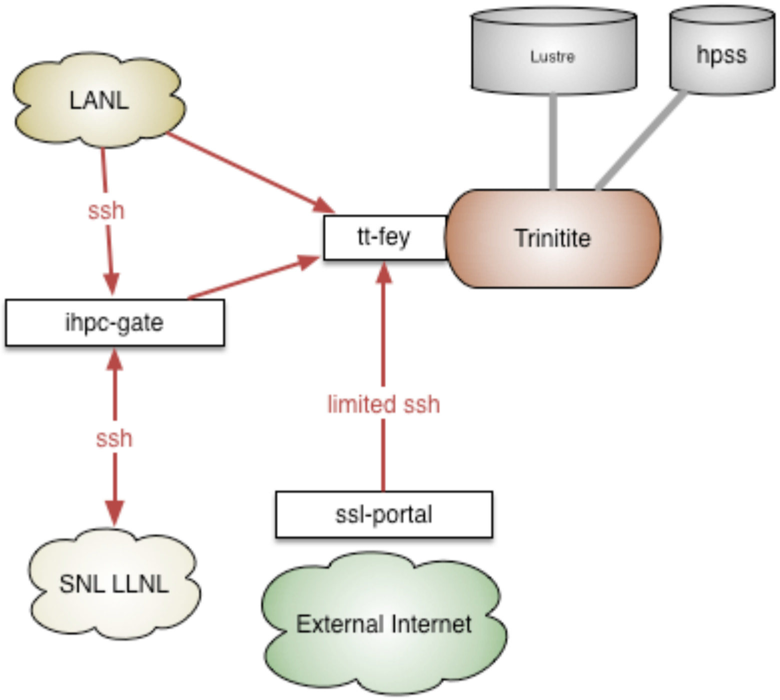
- Allocations
  - Apply for an ATS Campaign
  - Allocations follow the ATCC process: https://aces.sandia.gov/atcc.html
- Accounts requests
  - Trinity
    - LLNL & Sandia users: through SARAPE http://sarape.sandia.gov
    - LANL users: https://hpcaccounts.lanl.gov/
  - Trinitite
    - LLNL & Sandia users: through SARAPE http://sarape.sandia.gov
    - LANL users: https://hpcaccounts.lanl.gov/
  - Mutrino
    - LLNL & LANL users: through SARAPE http://sarape.sandia.gov
    - Sandia users: through https://webcars.sandia.gov

Accessing Trinity

- Start from a local gateway (e.g. `cslic.llnl.gov`, `lynx-s.sandia.gov`)
- Get ticket: `kinit -f`
- Use `ssh` to reach LANL gateway: `red-wtrw.lanl.gov`
- From there, `ssh` into Trinity front-end: `tr-fe` or File Transfer Agents (FTAs): `rfta-fe`



Accessing Trinity Application Regression Testbed (ART) Systems



Log in to Mutrino via srngate.sandia.gov

Diagram will be provided in published slides and on web pages

Trinitite

Mutrino

Scheduling and Jobs

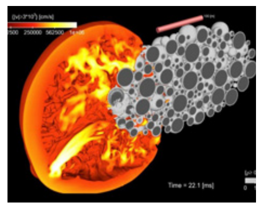
Job Scheduling

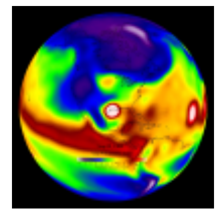
- Moab/Torque/ALPS
  - λ Same commands on all ASC platforms at all three Labs: `msub`, `showq`, `checkjob`, etc.
- Fairshare-based scheduling, 32-day history, no "bank", no upper-limit on total CPU time per Campaign

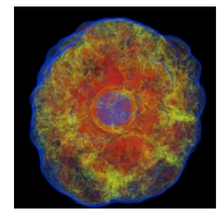
Running Jobs

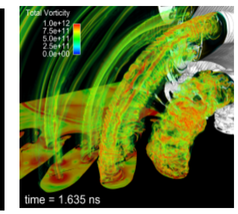
- `aprun -n` … (for default MPI)
- `mpirun -np` … (for Open MPI apps)
- Modulefiles software environment

Trinity Road Show
Programming Environment and Tools









Mahesh Rajan, Kevin Pedretti,
Jennifer Green, David Shrader

August 2015

Unclassified Unlimited Release: SNL Sand No: SAND2015-6926 PE

Trinity Programming Environment

Compute Node Linux (CNL) with LWK on Compute Nodes and Full SUSE Linux on Login Nodes

- Programming Languages
  - Fortran
  - C
  - C++
  - Python
  - Chapel
- Compilers
  - Cray Compiling Environment (CCE)
  - Intel
  - GNU
- Programming Models (Distributed Memory)
  - MPI: Cray Mpich & OpenMPI
  - SHMEM
  - OpenMP4.0
  - PGAS & Global View
  - UPC
  - CAF
  - Chapel
- I/O libraries
  - NetCDF
  - HDF5

- Tools
  - Use modules to setup software env.
- Debuggers
  - DDT
  - TotalView
  - Abnormal Termination Processing (ATP)
  - STAT
  - Cray Comparative Debugger
- Performance Analysis
  - Vtune and Vector Advisor
  - CrayPat and Cray Apprentice2
- Optimized Scientific Libraries
  - Libsci
  - MKL
  - LAPACK
  - ScalAPACK
  - BLAS
  - Cray Adaptive FFTs (CRAFFT)
  - Cray PETSc
  - Cray Trilinos
  - FFTW

Modules and Compiling

- Modules tool used to control needed environment variable and directories in your path
  - *module list* shows default
    - OS modules, PrgEnv Modules, and support modules
  - Use *module avail* to look for choices
  - Use “unload”, “load”, “swap” module commands to set it to your preference
- A special *“PrgEnv”* module controls compiler, debugger, and misc library setup. Select from:
  - *PrgEnv-intel; PrgEnv-cray; PrgEnv-gnu; PrgEnv-pgi;*
- Default compute node compiler wrapper commands for *all* compilers (same as Cielo):
  - *ftn, cc, and CC*

Running Applications MOAB/TORQUE/ALPS (same as Cielo)

- ALPS is the allocator provided by Cray for the compute nodes.
  - Works in concert with Moab and Torque
  - Provide batch job submission and job script launch facilities
  - Commands within ALPS: basic commands
    - running (*aprun*)
    - status enquiry (*apstat*)
    - termination (*apkill*)
- Useful man pages
  - *intro_alps*
  - *aprun*
- Use the *aprun* command and the $OMP_NUM_THREADS environment variable to run a hybrid program. You may need *aprun* options "-*n*", "-*N*", and "-*d*" to get the desired combination of MPI processes, nodes, and cores.
  - *aprun -n mpi_procs -N mpi_procs_per_node -d threads_per_mpi_proc mpi_program*
- Very good Cray documentation at http://docs.cray.com
  - Cray Application Developer's Environment User's Guide

Major Differences between Cielo and Trinity

	Cielo	Trinity (Phase-1)
Processor	AMD Opteron 6100 (MagnyCours)	Intel Xeon E5-2698V3 (Haswell)
nodes, cores/node, memory/node	8518 compute, 16cores/node, 32GB/node	9436 compute, 32cores/node, 128GB/node
Interconnect	Cray Gemini (3D Torus)	Cray Aries(Dragonfly)
Debuggers	TotalView	DDT and TotalView
Compiler (Default)	PGI	Intel
Scientific Library	libsci, ACML, MKL	libsci, MKL
SIMD	SSE4a	AVX2 with FMA
Peak Node GFLOPS	153.6	972.8 at base Turbo Freq. of 1.9GHz

Compilers and flags

	Intel	Cray	GNU	PGI
Optimization	-fast** –no-ipo	-O3 –h scalar3	-O3 –ffast-math	-fastsse
AVX	-xcore-avx2	-h vector3	-mavx	-Mvect=simd:256
OpenMP	-qopenmp*	-h omp	-f openmp	-mp=nonuma
Fortran Format	-free	-ffree	-ffree-form	-Mfree
Array bounds check	-check bounds	-h bounds	-fbounds-check	-Mbounds
Report	-Wextra -Wall	-h negmsgs	-fopt-info-vec	-Minfo=all

*Intel v15 and greater, otherwise –openmp

**-fast includes –fp-model fast=2: for tighter accuracy add –fp-model fast=1 or fp-model precise

Going from Cielo PGI to Trinity Intel

PGI	Intel	Description
`-fast`	`-fast -no-ipo`	Standard optimization
`-mp=nonuma`	`-qopenmp*`	OpenMP support
`-Mfixed`	`-fixed`	Fortran fixed format support
`-Mfree`	`-free`	Fortran free format support
`-byteswapio`	`-convert big_endian`	Read and write Fortran unformatted data files as big endian.
`Explicit link to mkl libs`	`-mkl`	Link to Intel MKL
`-V`	`--version`	Show compiler version

*Intel v15 and greater, otherwise -openmp

PGAS (Partitioned Global Address Space) languages: Unified Parallel C (UPC) and Coarray Fortran (CAF)

- PrgEnv-cray module provides native support for Coarray Fortran and Unified Parallel C:
- CAF example:
  - ftn -h caf -o CAFhello CAFhello.f90
- UPC example:
  - cc -h upc -o UPCProg UPCProg.c

Chapel (an emerging parallel programming language being developed by Cray Inc.) is also available through a module

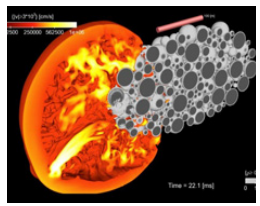
RUR - Resource Utilization Reporting (New in Trinity)

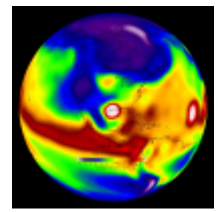
- Measures resources used by each application
- Users optionally receive report written to their home directory, appended to ~/rur.out
- Available Metrics
  - Timestamps: Start and end time of the application
  - Taskstats: User CPU time, max memory used, exit code, …
  - Energy: Total energy used by the application
- Example:

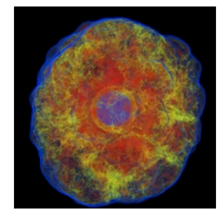
```
uid: 20983, apid: 175244, jobid: 349.mutrino-moab,
cmdname: ./bin/mpicth_244, plugin: energy
{"nodes_throttled": 2, "nodes_with_changed_power_cap": 0,
"max_power_cap_count": 0, "energy_used": 25015991,
"max_power_cap": 0, "nodes_accel_power_capped": 0,
"min_power_cap": 0, "min_power_cap_count": 0,
"nodes_power_capped": 0, "nodes": 96}
```

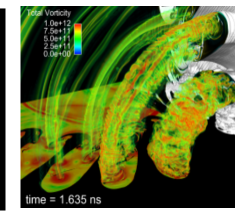
Trinity Burst Buffer

Hardware, Software and Usage









Cornell Wright
Nathan Hjelm

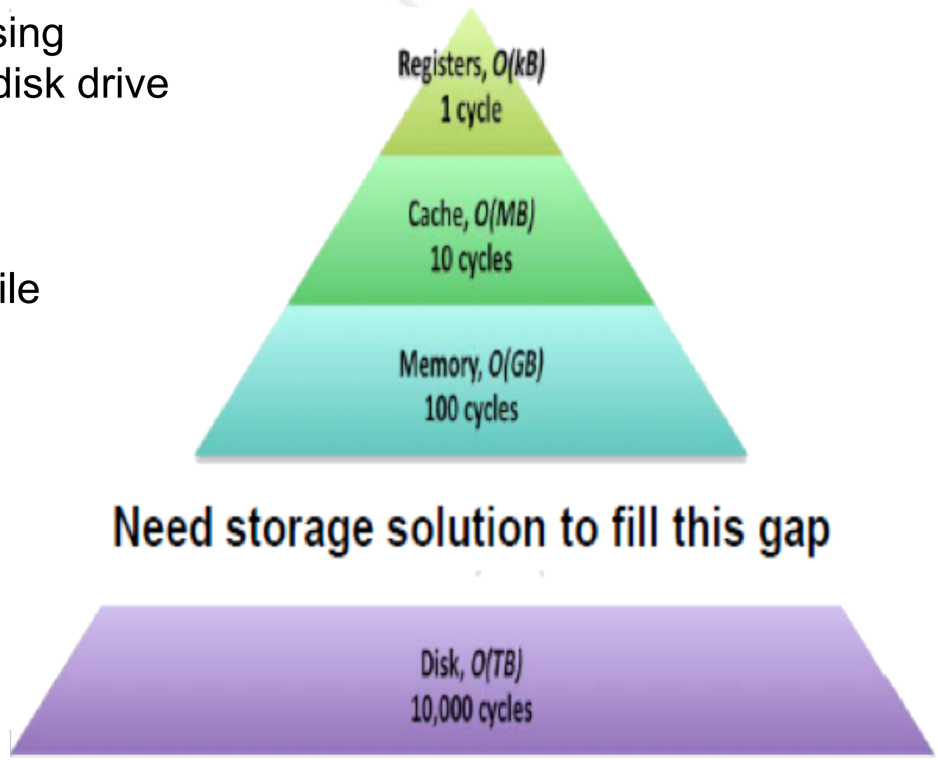
August 2015

UNCLASSIFIED - LA-UR-15-26608

Burst Buffers will improve Productivity and Enable Memory Hierarchy Research

Los Alamos NATIONAL LABORATORY EST. 1943

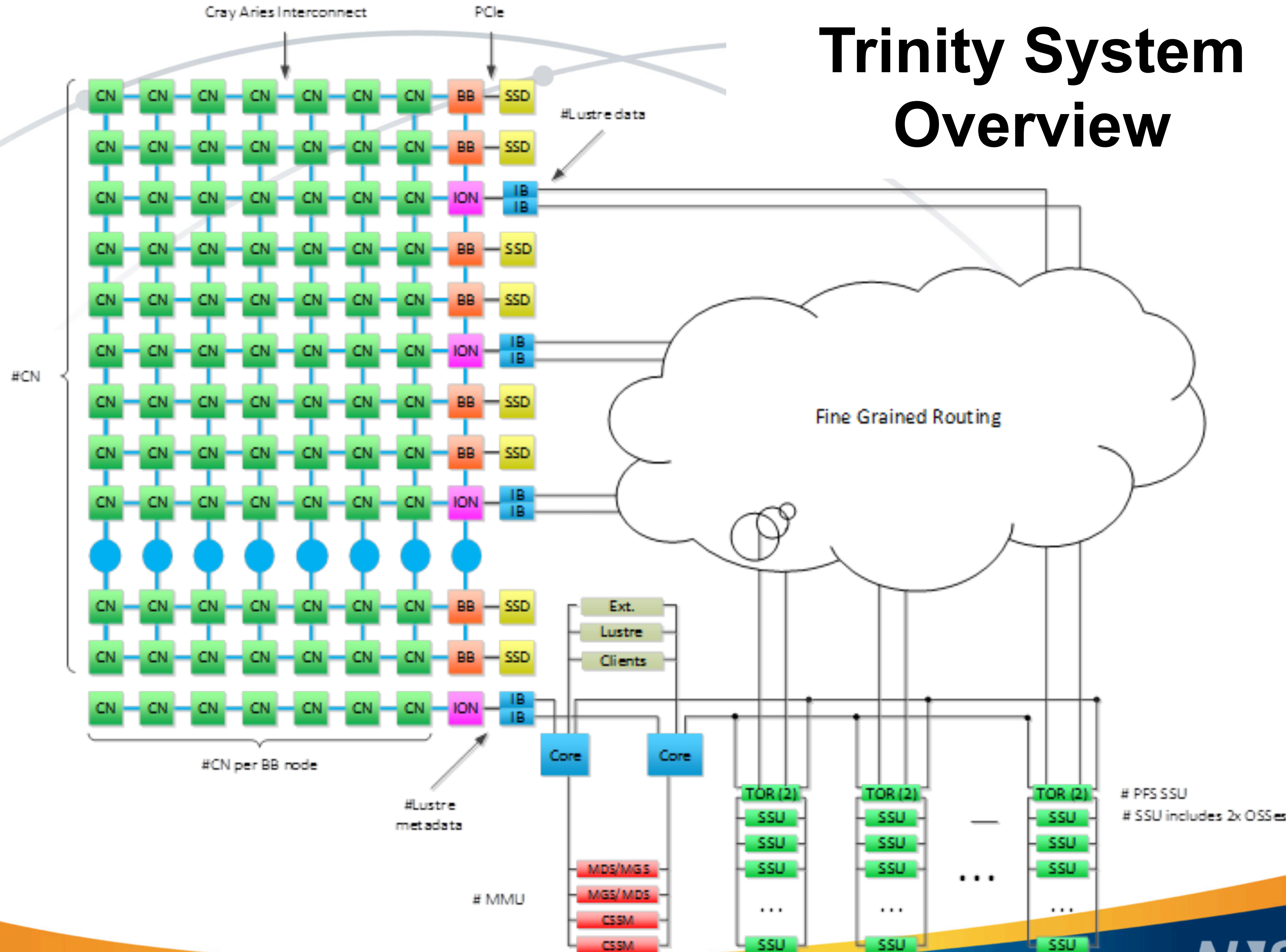
- Technology Drivers:
  - Solid State Disk (SSD) cost decreasing
  - Lower cost of bandwidth than hard disk drive
- Trinity Operational Plans:
  - SSD based 3 PB Burst Buffer
  - 3.28 TB/Sec (2x speed of Parallel File System)
- Burst Buffer will improve operational efficiency by reducing defensive IO time
- Burst Buffer fills a gap in the Memory and Storage Hierarchy and enables research into related programming models



Burst Buffer – more than checkpoint

- Use Cases:
  - Checkpoint
    - In-job drain, pre-job stage, post-job drain
  - Data analysis and visualization
    - In-transit
    - Post-processing
    - Ensembles of data
  - Data Cache
    - Demand load
    - Data staged
  - Out of core data
    - Data intensive workloads that exceed memory capacity

Trinity System Overview



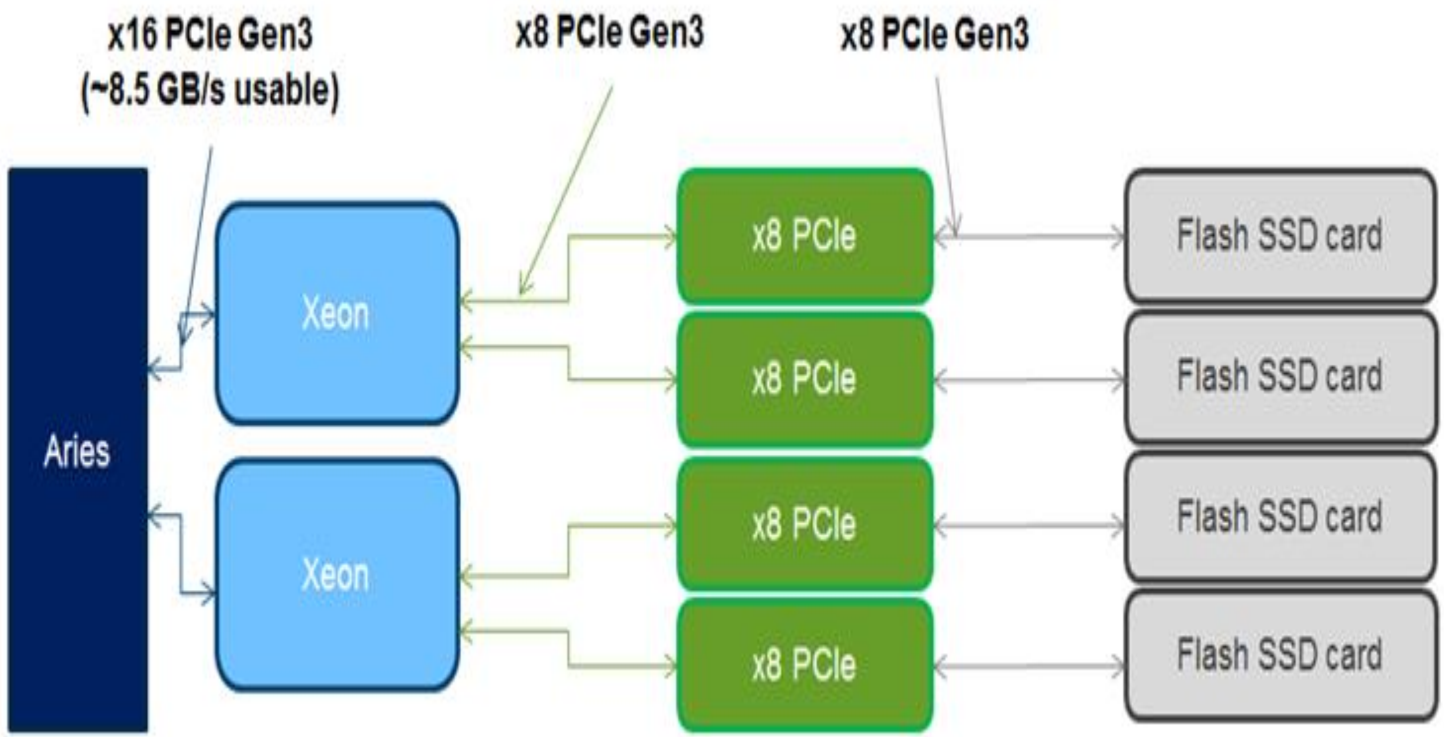
Trinity Burst Buffer Hardware

- 576 Burst Buffer Nodes
  - Announced as Cray DataWarp™
  - On high speed interconnect - globally accessible
  - Trinity IO Node + PCIe SSD Cards
  - Distributed throughout cabinets

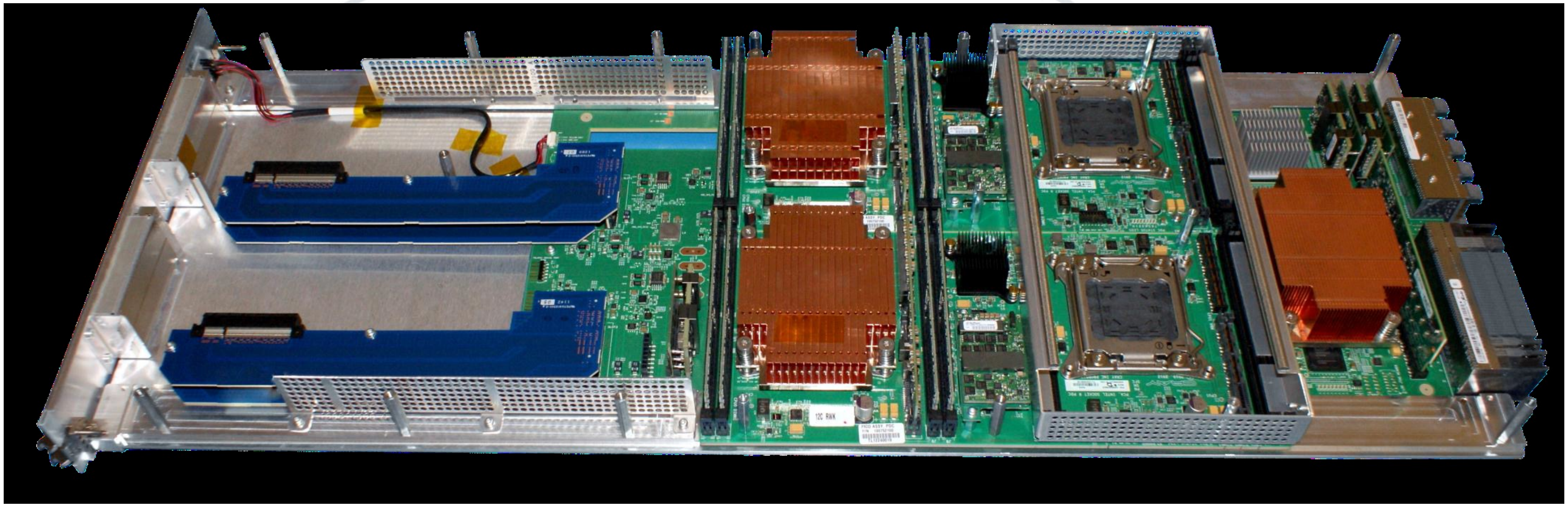
Metric	Phase I		Full Trinity	
	Burst Buffer	PFS	Burst Buffer	PFS
Nodes	300 BB Nodes	114 LNET Routers	576 BB Nodes	234 LNET Routers
Bandwidth	1.7 TB/S	0.71 TB/S	3.3 TB/S	1.45 TB/S
Capacity	1.9 PB	82 PB	3.7 PB	82 PB
Memory Multiple	1.6 X	67 X	1.75 X	39 X
Full system checkpoint cost			12%	21%

Cray XC40 DataWarp Blade

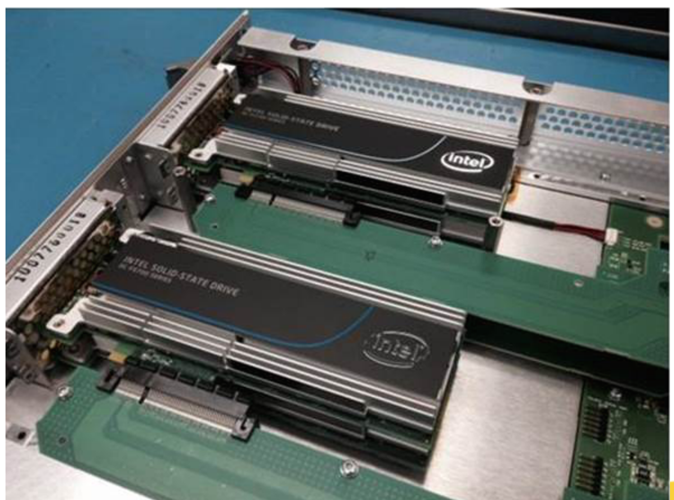
(2 Burst buffer Nodes)



Cray DataWarp Blade



↑ Service Blade (2 nodes)



← SSD Cards

Burst Buffer Operating Modes

Mode	Description
Private Scratch	Per node burst buffer (BB) space
Shared Scratch	Shared space, files may be striped across all BB nodes. → Used for Trinity Checkpoints ←
Shared Cache	Parallel File System (PFS) cache. Transparent and explicit options
Load Balanced Read Only Cache	PFS files replicated into multiple BB nodes to speed up widely read files

Burst Buffer System Software

- Burst Buffer SSD partitioned into allocation units
  - Allocation units belong to LVM volume group
- Workload manager
  - Job submission requests BB capacity
  - Starts job when capacity available
- DataWarp registration service
  - Selects allocation units
  - Creates XFS logical volumes on SSD
  - Mounts via DVS on compute nodes
- Multiple Access Modes
  - Scratch / Cache
  - Striped / Private / Load Balanced (RO)
- Trinity BB Checkpoints will use Striped Scratch
- Automated stage/drain of specified directories from/to PFS
- Per job write limits (endurance management)
- Administrative Functions – configuration, monitoring, repair

DataWarp Job Script – Access

- Jobdw
  - Create and configure a DataWarp job instance
  - #DW jobdw [access_mode=mode] [capacity=n] [max_mds=n] [modified_threshold=bytes] [pfs=path] [read_ahead=bytes:rasize] [sync_on_close=yes|no] [sync_to_pfs=yes|no] [type=cache|scratch]
- Persistentdw
  - Configure access to an existing persistent DataWarp instance
  - #DW persistentdw name=piname

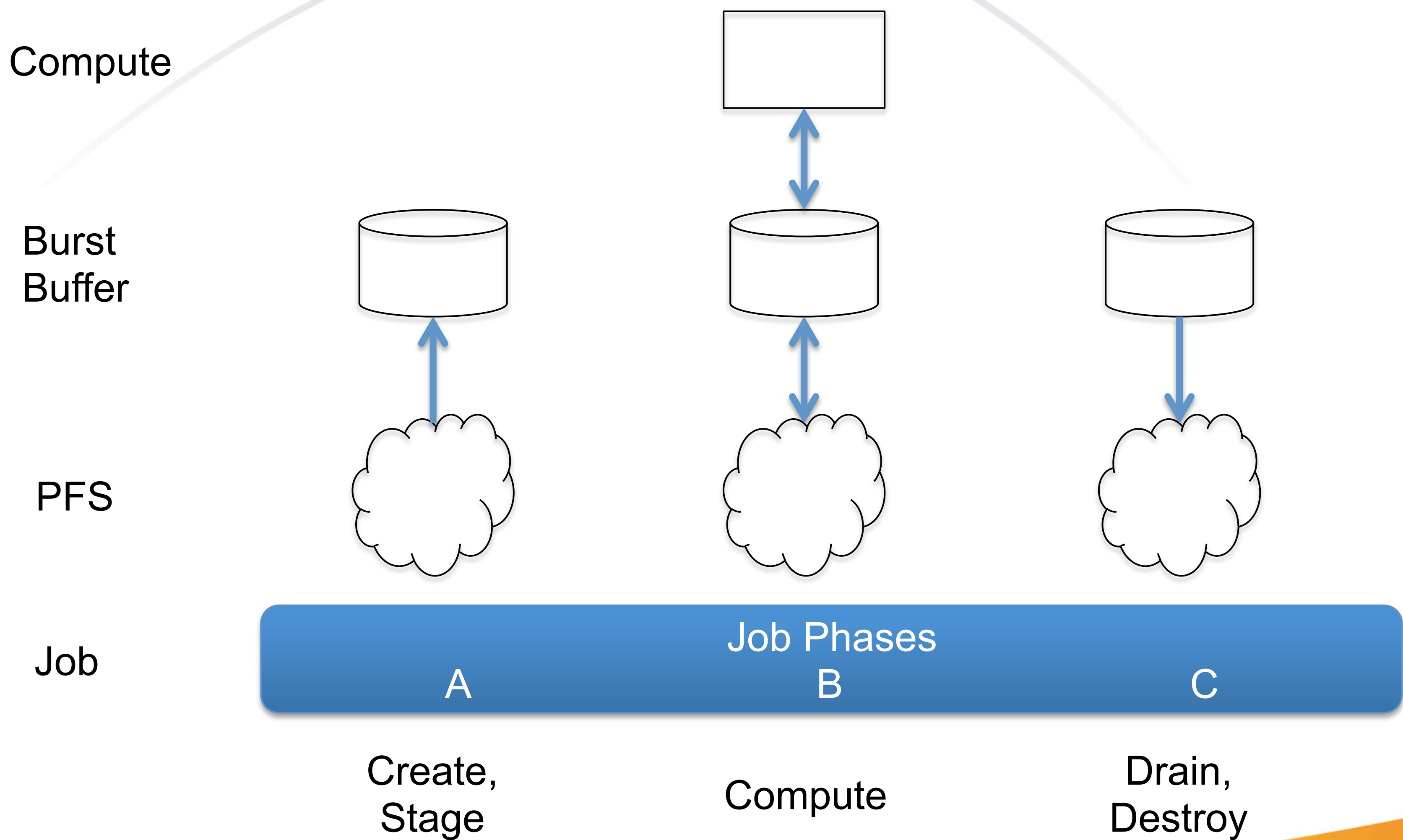
DataWarp Job Script – Staging

- stage_in
  - Stage files into a DataWarp instance
  - #DW stage_in destination=dpath source=spath [tolerate_errors=yes|no] type=type
- stage_out
  - Stage files from a DataWarp instance
  - #DW stage_out destination=dpath source=spath type=type

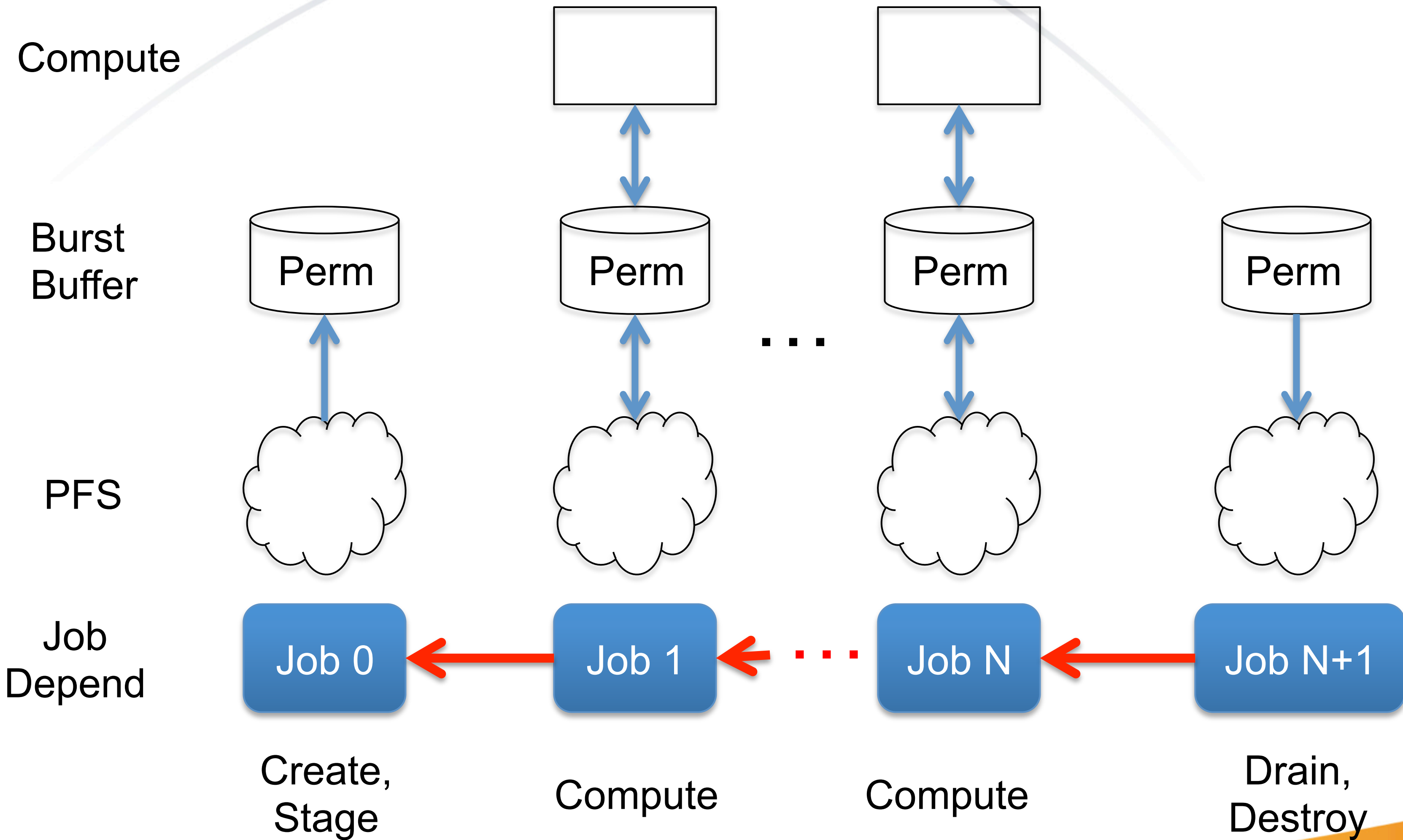
DataWarp Job Script – Swap

- Swap
  - Configure swap space per compute node
  - #DW swap nGiB

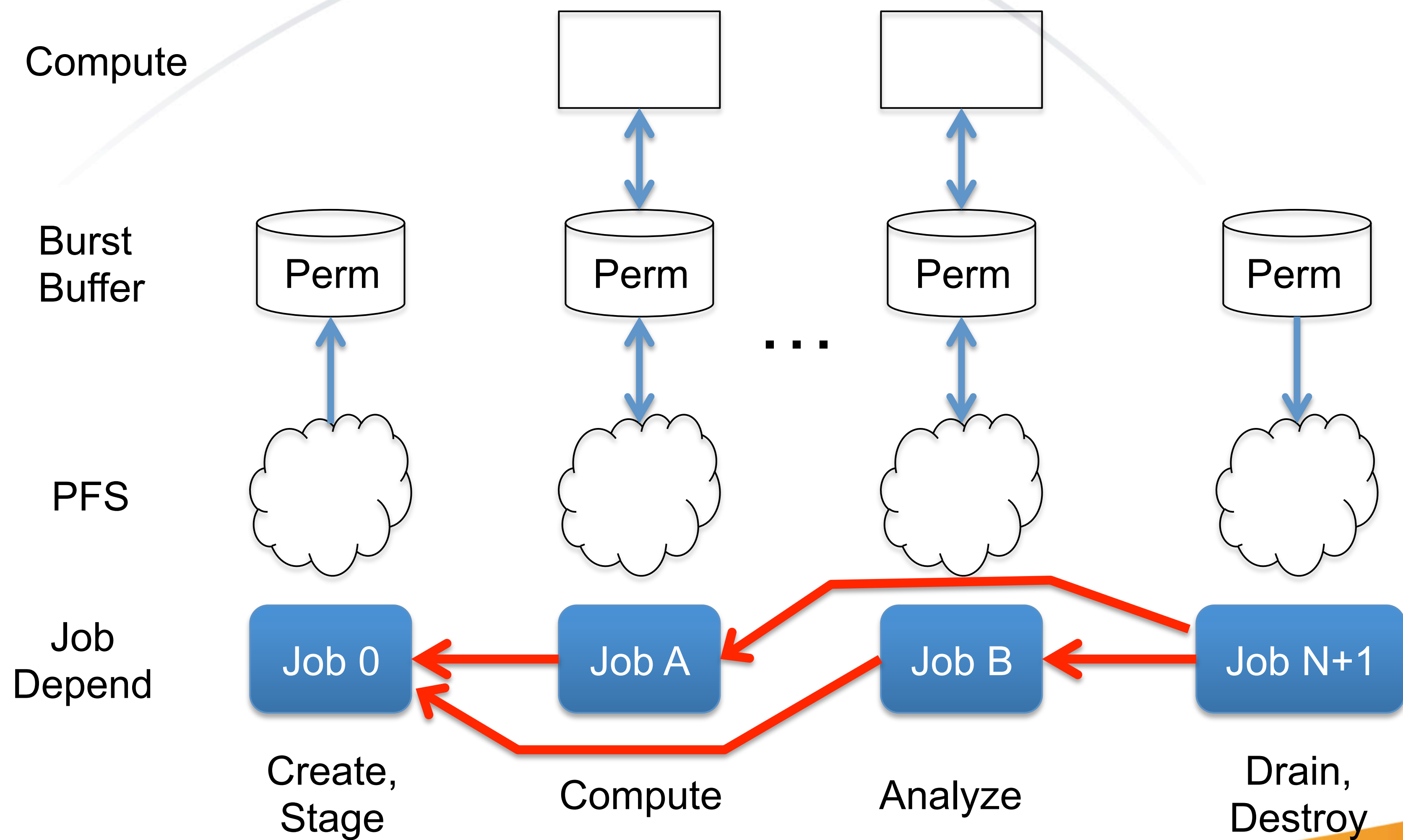
Single-Job Burst Buffer



Multi-Job Burst Buffer



Compute + Analysis Burst Buffer



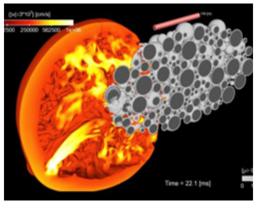
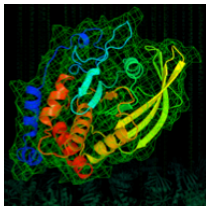
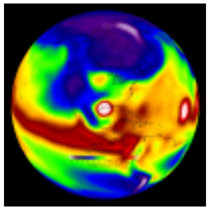
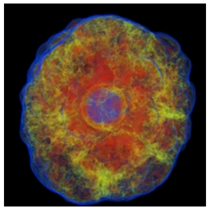
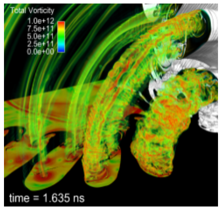
Applications have Options

- Hierarchical Input Output (HIO) Library
  - Hides vendor specific interface
  - Provides additional performance, reliability, management and diagnostic functions
  - Recommended approach
- Scalable Checkpoint Restart Library (SCR)
  - Adam Moody (LLNL) planning to support Trinity BB with SCR
- Direct POSIX calls from application
  - Will require Cray DataWarp specific ioctl calls to exploit striping and stage / drain functions

Acknowledgements

- Many people and groups have contributed to the concept, design and development of burst buffer, Trinity and the materials in this talk.
- Including:
  - Gary Grider
  - Josip Loncaric
  - Doug Doerfler (SNL)
  - Nathan Hjelm
  - Nick Wright (NERSC)
  - Dave Henseler (Cray)
  - Bob Pearson (Cray)
  - Bronis de Supinski (LLNL)
  - Adam Moody (LLNL)
  - John Bent (EMC)

Trinity Platform Application Readiness

Application Readiness
Cornell Wright, Joel Stevenson

August 2015

Application Readiness Objectives

- Understand application/user requirements, usage patterns, and workflows
- Be an advocate for customer needs
- Work with application teams to ensure production-run readiness on current and incoming systems
- Engage with Trinity Early Access codes to achieve rapid, productive use
- Identify causes of unexpected behavior; submit bugs and track fixes
- Collaborate with subsystem teams such as Systems Management, File Systems, I/O, Archive and Tools
- Stress and regression test of production and new systems and system configurations

Application Readiness Approach

- Each participating site (LANL, LLNL, and SNL) has some form of a local application readiness team.
- Local application readiness teams will work with code developers at their sites to port codes from Cielo to Trinity.
- ACES application/user support personnel have the responsibility for working with site local application readiness teams and their code developers to port codes to Trinity Phase 1 (Haswell). We call this partnership the "ACES Application Readiness Team".
- The Trinity Application Center of Excellence (COE) has the responsibility for working with site local application readiness teams and their code developers to port selected codes to Trinity Phase 2 (KNL).

ACES Application Readiness Team

- ACES application/user support experts:
  - Consult at LANL
  - OneStop at SNL
- LANL AR team:
  - Cornell Wright
  - Dave Nystrom
  - David Gunter
  - Nathan Hjelm
  - Howard Pritchard
  - Mike Berry (Cray)
- SNL AR team:
  - Joel Stevenson
  - Karen Haskell
  - Dennis Dinge
  - Mike Davis (Cray)
- LLNL Development Environment group:
  - Scott Futral

Trinity Phase 1 (Haswell)

- Codes running on Cielo need to run well on Trinity Phase 1 from the beginning of General Availability.
- The ACES Application Readiness Team will work with code teams to port/test the codes and evaluate their production readiness.
- The ACES Application Readiness Team will also work closely with the Campaign users to exercise the codes with user workflows and evaluate the performance of the platform/codes on the user Campaign workload.

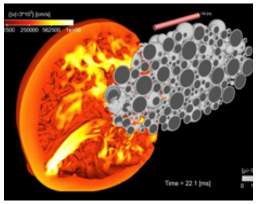
Trinity Phase 2 (KNL)

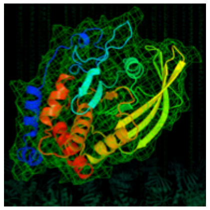
- The first application codes to be ported to Trinity Phase 2 have been identified and will be managed by the Trinity Application Center of Excellence (COE) activities. Three codes have been selected, one from each lab.
- Additional codes will be selected for COE efforts based on ASC programmatic needs. ASC resources are limited, so attention will focus on a limited number of applications in the beginning.
- Before becoming Trinity Phase 2, application codes will need to demonstrate success on the KNL architecture. Success on the KNL architecture may be demonstrated in several ways:
  - Work with Trinity COE to insure the code is ready for Trinity KNL usage.
  - Demonstrate on Trinity KNL architecture testbed(s) that the code uses the hardware efficiently and is ready for Trinity KNL usage. Then the code may move to the ACES Application Readiness Team for adoption in Trinity Phase 2.
  - Demonstrate on Trinity KNL that the code uses the hardware efficiently and is ready for Trinity KNL usage. Then the code may move to the ACES Application Readiness Team for adoption in Trinity Phase 2.

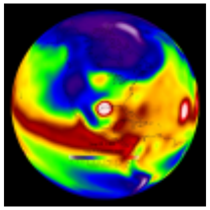
Application Readiness Process

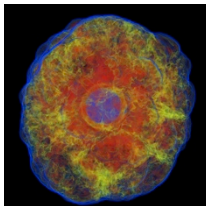
- Engage consultants, web documents, etc. for normal needs. When this is not sufficient or not working . . .
  - Los Alamos – contact Cornell Wright, cornell@lanl.gov
  - Sandia – contact Joel Stevenson, josteve@sandia.gov
  - Livermore – contact Scott Futral, futral2@llnl.gov
- Trinity Applications Technical Concall
  - Wednesdays 3 MT / 2 PT
  - LANL 6-1220 or 866-880-6515 PC 043534 #
- Issues/problems that are primarily system related can go directly to the Los Alamos AR team

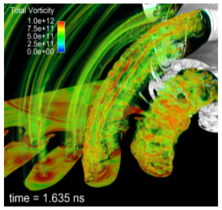
Trinity Campaign Storage and Usage Model











Kyle Lamb

August 2015

Campaign Storage Topics

- Who
- Why
- How
- Where
- When

Campaign Storage Thanks

- Many Thanks go to:
  - HPC-3 Storage
  - HPC-5 Storage
  - Vendors

LANL Archive Performance

- Restore of a 25TB file = 30Hours
- File sizes expected in the 100s of TB for Trinity
- Need to filter data going to archive, archive is for forever data (code repos, input decks, some visualizations) not necessarily checkpoints
- Need to start planning for 1PB sized checkpoints, with longer residence with high performance access

Campaign Storage Why

- Long term high performance data storage repository for large data sets
- Store data for the duration of a computing campaign on a storage tier that offers higher resilience than Scratch and higher performance than Archive
- At completion of campaign, user selection is made to determine what needs to be archived

Campaign Storage Specifications

- Capacity ~25PB (with future expansion expected)
- Performance ~20-25GB/s (expansion will increase performance)
- Quotas enabled
- Full file system view (ls and grep work)
- No update in place

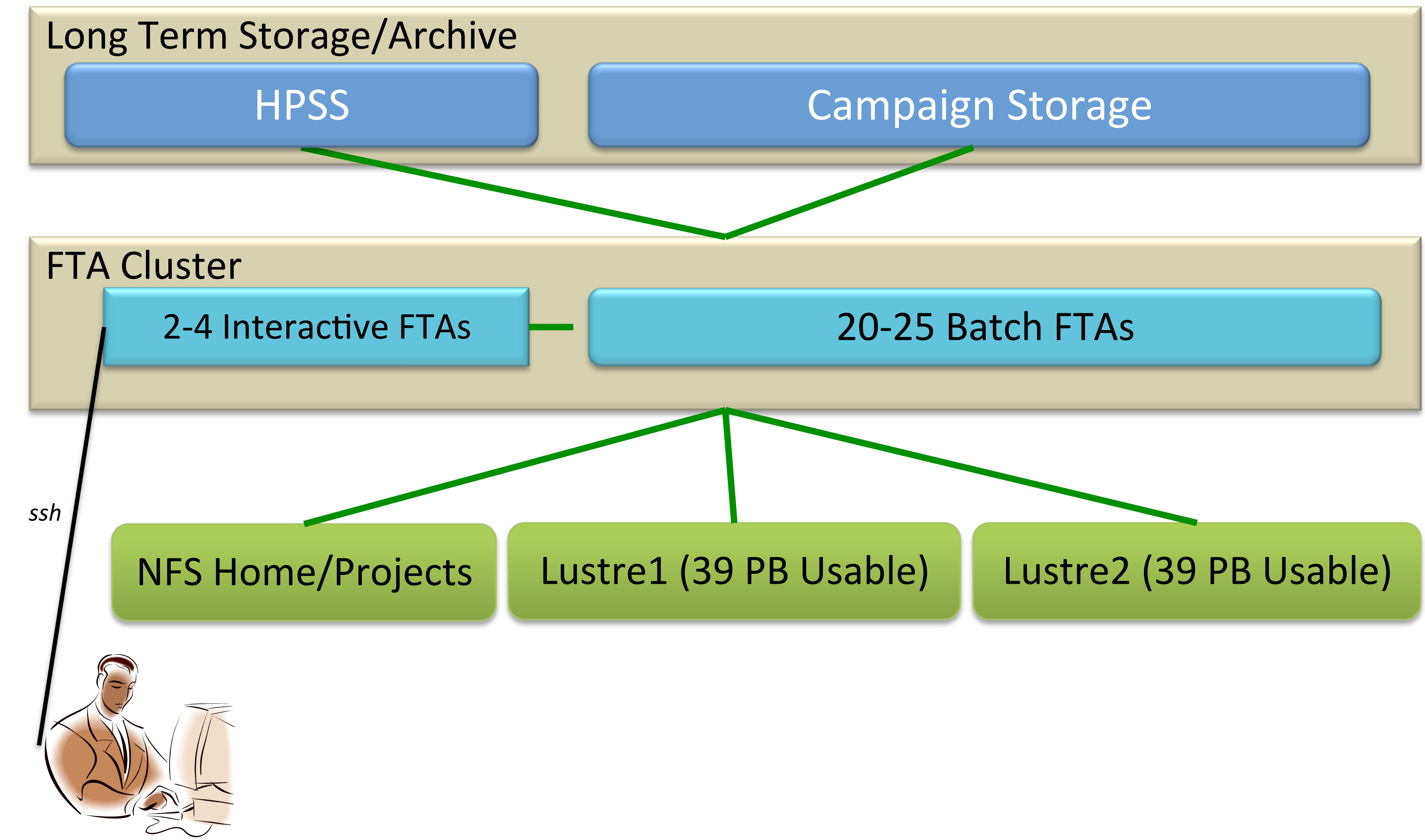
Campaign Storage Technology

- Posix file system with Object storage backend
- Utilizes “cloud driven” erasure code to gain highly reliable disk based storage (20+4, 30+6, 40+8)
- Higher performance gained through large scale parallel access to archive grade hard drives
- Not intended for high duty cycle workloads e.g. Scratch

Campaign Storage Access

- Available only via the FTAs not mounted to any cluster
- Mounted on interactive FTAs but performance available via batch FTAs and PFTOOL *new*
- FTAs will mount all production file systems and utilize HSI for interface with HPSS

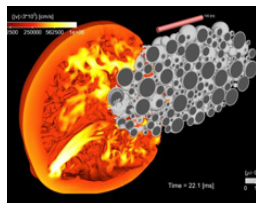
Campaign Storage Diagram

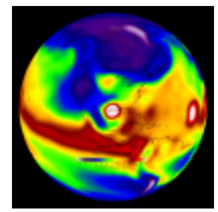
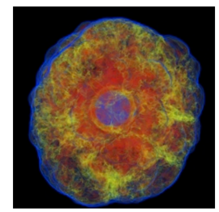
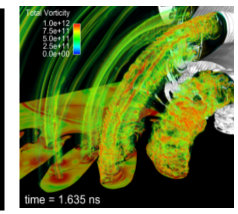


Campaign Storage Timeline

- Small 3PB version will be released with Trinity for Open Science
- Larger 25PB version available when Trinity goes into production
- Capacity and Performance increases will occur with need and available budget

Visualization on Trinity



Visualization on Trinity

Laura Monroe, David Modl, Robert Kares, John Patchett (LANL)
David Karelitz, Ken Moreland, Warren Hunt)SNL)
Eric Brugger (LLNL)

August 2015

Viz Usage Scenarios on Trinity:
Post-processing, off-platform rendering

- Interactive post-processing with off-platform rendering
  - Dedicated viz nodes located within the platform run data servers that read data from the global PFS and extract visible geometry.
  - The visible geometry is sent through the GigE backbone over to local clients for hardware rendering.
  - Standard approach used with Cielo and previous ASC platforms.
  - A low risk option: EnSight data servers will run out of the box on the dedicated viz nodes on the Haswell side of Trinity.

Viz Usage Scenarios on Trinity
Post-processing on-platform rendering

- Interactive post-processing with on-platform rendering
  - Dedicated viz nodes located within the platform run data servers that read data from the global PFS and extract visible geometry.
  - The visible geometry is passed to clients running on the dedicated viz nodes that render on the node CPU's using a Mesa 3D interface to Intel's OpenSWR renderer.
  - New option available with Trinity.
  - Requires successful development by Intel of their OpenSWR renderer for Haswell and KNL and deployment of the renderer into mesa 3D.
  - OpenSWR development for Haswell is relatively complete.
  - KNL support for OpenSWR is under development and is higher risk.
  - EnSight, ParaView and VisIt all plan to support this option.

Viz Usage Scenarios on Trinity
In-transit visualization using burst buffer

- Co-processing for in-transit viz with the burst buffer
  - In this scenario dedicated viz nodes located within the platform share access with the compute nodes of a running simulation job to a burst buffer filesystem.
  - The simulation job writes viz data to the burst buffer and data servers running on the viz nodes read data from the burst buffer and extract visible geometry.
  - The visible geometry can be rendered either on or off platform.
  - This type of co-processing activity is termed in-transit viz.
  - Can be used for both interactive and batch visualization.

Viz Usage Scenarios on Trinity:
In-situ visualization

- In-Situ viz using ParaView/Catalyst or VisIt/LibSim in simulation codes
  - Simulation code is linked with the ParaView/Catalyst or VisIt/LibSim libraries to allow complete visualization pipelines to be executed at runtime in the simulation memory space.
  - Visualization pipelines are defined by Python scripts read at runtime by ParaView/Catalyst or VisIt/LibSim to produce data products such as imagery, vis dumps, or derived data products
  - Requires integration with the simulation code source to pass control to ParaView/Catalyst or VisIt/LibSim and to link with the analysis libraries.
  - This approach has recently been successfully tested on Cielo using ParaView/Catalyst on the RAGE code.

Dedicated Viz Nodes for Trinity

- For Trinity we intend to continue the viz model very successfully utilized for Cielo and most previous generations of ASC machines.
- 2% of Trinity nodes will be dedicated *exclusively* for visualization and analysis.
  - The partition of dedicated viz nodes will be on both the Haswell and KNL sides of the machine.
  - Percentage based on observed Cielo use
- Percentage reviewed every ATCC cycle, and readjusted if needed
- Can request extra resources to handle special needs
  - Nodes should support dynamic linking to shared object libraries,TCP/IP sockets for internal and external communication, DISCOM

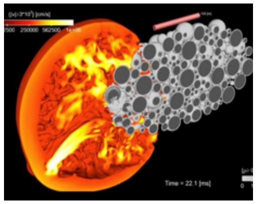
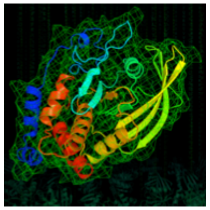
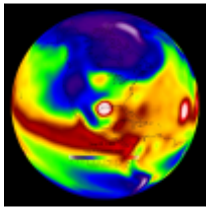
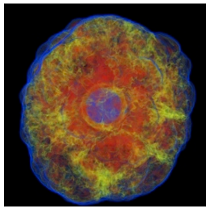
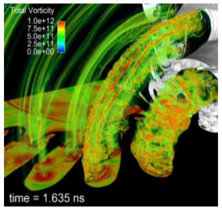
Rendering on KNLs

- Rendering issue on Intel Xeon Phi's
  - OpenGL libraries not updated on this platform
  - OpenGL runs slowly
- Impact to Trinity and Cori platforms
  - In particular impacts in-situ rendering
- Intel plans:
  - Development on Intel's open source OpenSWR and deployment of the core renderer into Mesa 3D, providing a Mesa interface to the codes
  - Provide analysis/proposal/commitment to address this
  - Develop and deploy software onto KNL
- Lab plans: working group
  - Review proposal
  - Work with Intel to develop and deploy Mesa/OpenSWR on Trinity and Cori
  - Representatives from LANL/SNL/LLNL/LBL
  - Lab people are EnSight/ParaView/VisIt developers and systems people

Viz Packages

- Ensight, ParaView, VisIt
- All three support:
  - OpenSWR/Mesa 3D support for on-platform rendering using Haswell and eventually KNL.
- EnSight and ParaView support:
  - Co-processing with the burst buffer for the in-transit viz usage scenario. (VisIt may support this)
- ParaView and VisIt support:
  - Batch production of images via in-situ viz for LANL, SNL and LLNL codes.

Archive and Data Transfer

Brett Hollander – LANL
Susie McRee - SNL

Archive in the Trinity Era

- HPSS will be the "forever" storage system.
  - Checkpoints will not be stored in HPSS under new storage model
  - HPC-3 recommends that single file store does not exceed 25TB
  - If user believes they have a need to archive large checkpoint files, meeting between HPC/Program management necessary. Exceptions will be few for files > 25TB
  - To ensure annual storage does not exceed budgeted capacity, quotas on LANL HPSS will be enforced.

Archive Quotas

- Why LANL HPSS quotas?
  - Must curb growth to sustain economically feasible Archive.
  - Out of 190 active users, the top 10 take up 45% of archive capacity ($2.5M worth of tape)
  - Only 5% recall rate
  - LANL HPC approached by ASC to implement

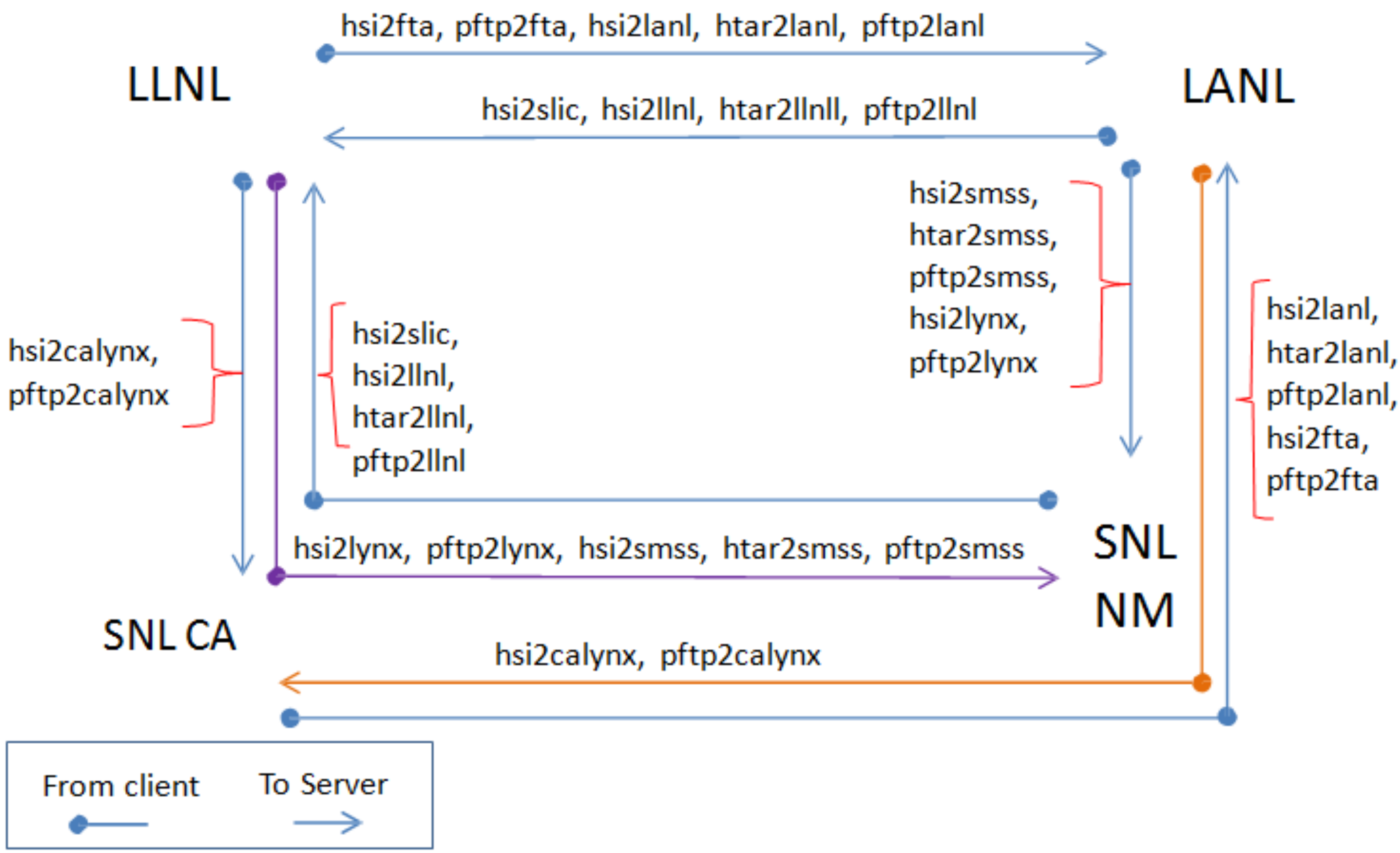
Archive Quotas Details

- Status: Proposed HPSS quotas in discussion with ASC
- "Soft" quotas, no automatic storage shutoff
- Quota reset each FY
- Amount of storage granted per user TBD
- User will receive notifications at 70%, 90%, 100%
- HPC+ASC management discussion takes place at 100%
- Management can request more storage in HPSS/Campaign or make determination that user must delete files or cease Archival storage
- Without an exception, if user continues to store past 100% their job priority will fall and HPC cluster accounts may get disabled

Archive Performance

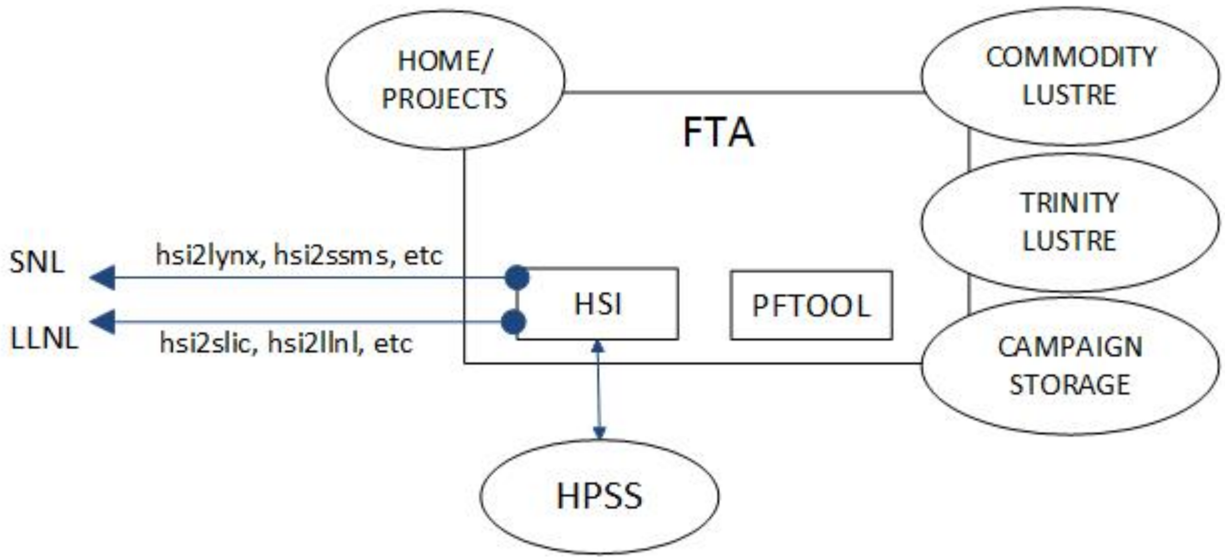
- HPSS will provide less performance than Campaign Storage
  - HPSS not designed for fast retrieval back to File system.
  - Single file/single node (300MB/s)
  - Multi-file/multi-node: (2-3GB/s)
  - Expect modest increase in performance by Trinity timeframe

Tri-lab Classified Data Transfer



LANL Internal Data Movement

- Use PFTool to…
  - Move data between Campaign Storage and Lustre
- Use HSI to…
  - Move data from Lustre/HPSS to SNL, LLNL
  - Move data between HPSS and Campaign
  - Move between Campaign and Lustre



Data Transfer Best Practices

- Use of HTAR is highly recommended for transfers of 1,000+ small files (< 1GB each).

- Use of HSI Transfer Agent will improve performance via parallel file transfer. It is recommended that files 1TB or larger be stored with HSI TA.

- Given that current DISCOM bandwidth maximum throughput is approximately 1TB/hour, transferring files > 10TB is not suggested.

Questions on any of the topics presented?

- For general Trinity information , see trinity.lanl.gov (public website)
- For detailed information about Trinity platforms, visit https://aces.sandia.gov/trinity (authentication required)

Backup Slides

Burst Buffer

Backup

HIO Design Goals

HIO, short for Hierarchical Input Output, is the burst buffer enablement library for Trinity and other systems.

Goals:

- Support Trinity and future Burst Buffer implementations
  - Isolate application from BB technology shifts
- Easy to incorporate into existing applications
  - Lightweight, simple to configure
- Improve checkpoint performance
- Improve job reliability and efficiency
- Support checkpoint and analysis scenarios (e.g., viz)
- Extend to bridge to future IO

Why implement HIO as a library ?

- Simplest packaging and delivery available
- Self contained, minimal external or system dependencies
- Easiest for applications to adopt
- Library approach facilitates rapid prototyping and deployment, responsiveness to application needs
- Library also provides a vehicle to provide (at no cost to applications):
  - Checkpointing best practices
  - Performance and functional diagnostics
  - Mitigation for system problems
- Why not provide via extensions to MPI-IO now ?
  - Existing implementation perform poorly
  - Unloved by users
- Integration with other IO packages will be investigated in future
  - HDF5 ? SCR ? ADIOS ?
  - HIO library should be largely reusable in that environment

HIO Project Features

- Full thread safety
- C/C++/Fortran support
- Configurable diagnostic and performance monitoring
- Header / Library packaging
- Open-source intention
- Support tri-lab ATS and CTS systems (more than Trinity)
- On-site (at LANL) prototype on Gadget
- Intent to prototype EAP support for HIO as POC and test vehicle
- PFS-only version available before Trinity

Primary HIO API Calls

- Init / Term / CP Mgmt:
  - hio_init_mpi()
  - hio_config_set()
  - hio_should_checkpoint()
  - hio_fini()
- Open / Close:
  - hio_dataset_open()
  - hio_element_open()
  - hio_element_close()
  - hio_dataset_close()
- Read / Write:
  - hio_element_write {strided}{nb}()
  - hio_element_read {strided}{nb}()
  - hio_wait()

hio – API calls Categorized

- **Context Management**
  - hio_init_single
  - hio_init_mpi
  - hio_fini
- **Checkpoint Management**
  - hio_dataset_should_checkpt
- **Data Collection Control**
  - hio_dataset_open
  - hio_dataset_close
  - hio_dataset_get_id
  - hio_dataset_unlink
  - hio_element_open
  - hio_element_close
  - hio_element_size
  - hio_dataset_construct
- **Data Movement**
  - hio_element_write
  - hio_element_write_nb
  - hio_element_write_strided
  - hio_element_write_strided_nb
  - hio_element_flush
  - hio_element_read
  - hio_element_read_nb
  - hio_element_read_strided
  - hio_element_read_strided_nb
  - hio_dataset_flush
- **Request Control**
  - hio_complete
  - hio_request_test
  - hio_request_wait
- **Configuration**
  - hio_config_set_value
  - hio_config_get_value
  - hio_config_get_count
  - hio_config_get_info
- **Performance Reporting**
  - hio_perf_get_count
  - hio_perf_get_info
  - hio_perf_get_value
- **Error Reporting**
  - hio_err_get_last
  - hio_err_print_last
  - hio_err_print_all

DataWarp Administrative Interfaces

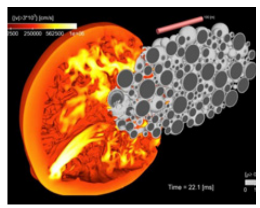
- Commands for setup and initial stage in
  - Issued by pre-compute system job based on #dw commands in job script
- Commands for final stage out and takedown
  - Issued by post-compute system job
- Monitoring for errors, endurance
- Commands for management – online / offline for repair, re-configuration

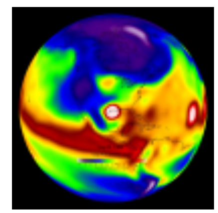
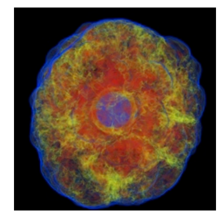
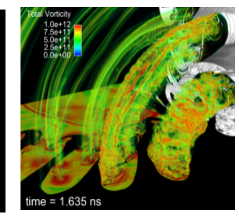
End of

Burst Buffer

Backup

Trinity Center of Excellence
SNL



Rob Hoekstra, Mike Glass, Ron Green(Intel) et al.

August 2015

SAND2015-6916 PE

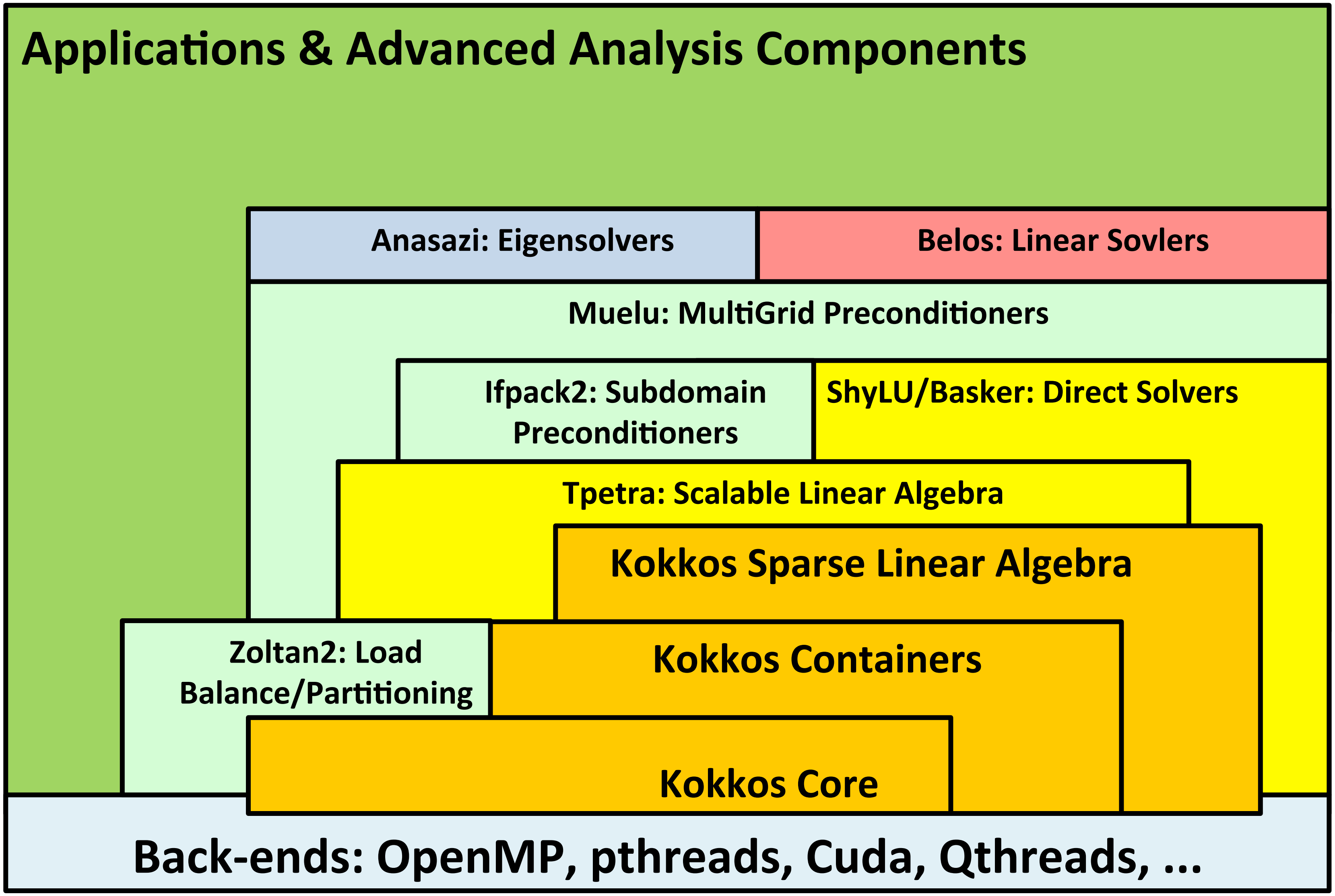
Priorities for CoE

- Proxy Applications
- Solvers
- Kokkos
- Codes
- Conclusions

Proxy Applications

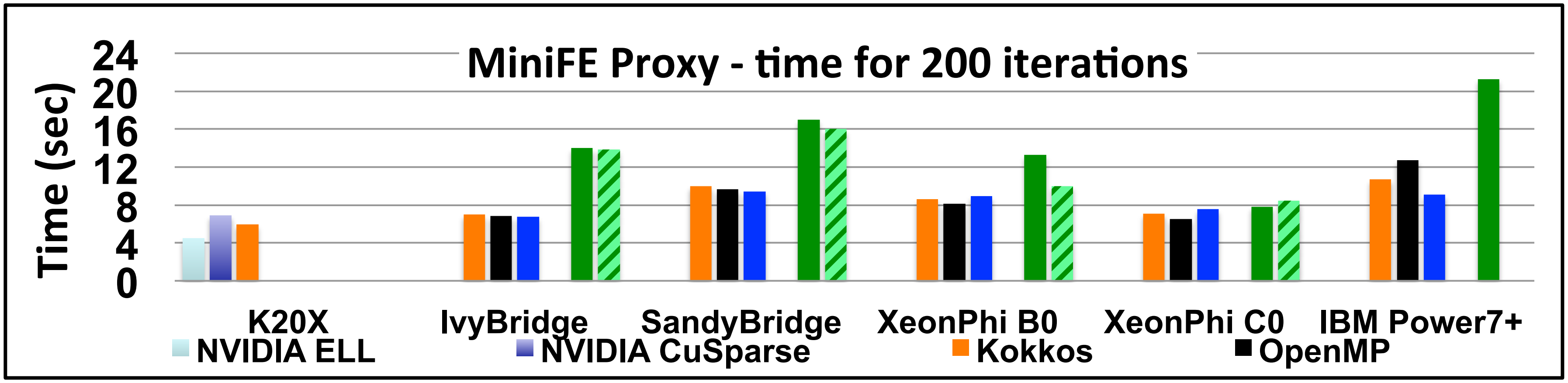
- miniAero
  - Explicit Aerodynamics
- miniFE/HPCG
  - Implicit Finite Element
- (mini)FENL
  - In progress, “miniDriver” more representative of real code data structures
- BDDC Domain Decomp. Solver Proxy

Trilinos NextGen Solver Stack



Kokkos

- Many-core challenge: “Refactor no more than 1+ε times!”
- In Use: Trilinos Solvers, Proxies, Science Applications



Application and Domain Specific Library Layer(s)

Kokkos Sparse Linear Algebra

Kokkos Containers

Kokkos Core

Back-ends: OpenMP, pthreads, Cuda, Qthreads, ...

Codes

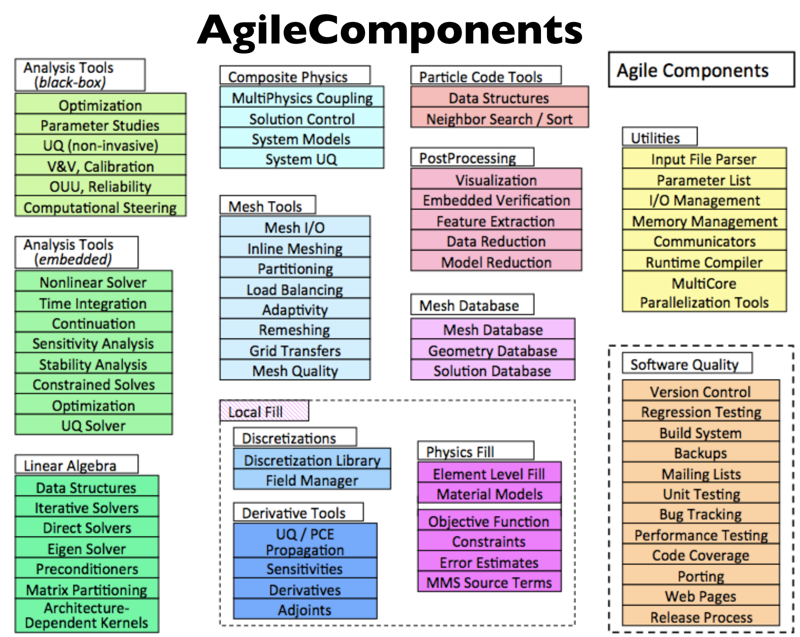
- SIERRA
  - Nalu (Science Code & Acceptance Test)
  - Solid Mechanics/Structural Dynamics
  - Aero
- RAMSES
  - Solvers, etc.
- ATDM
  - Dharma/Kokkos
  - Agile Components

SIERRA

- Solvers
  - Multi-grid (Nalu, full scale Cielo/Sequoia)
  - Domain Decomposition (vendor & custom subdomain solver
- Threading and Vectorization
  - (mini)FENL in progress
  - Intrinsics & Improved Intel Compiler
- Speciality Algorithms
  - Contact (exploring threading and novel new algorithmic approaches)

Agile Components

- **AgileComponents infrastructure**
  - **Defining a strategy for impact**
- **Deployed through algorithms products**
  - Trilinos – general algorithms toolkit
  - Dakota – UQ and optimization
- **Applications**
  - Sierra
  - RAMSES
  - Science Codes
  - ASCR



R&D 100 Award
1000s of Users Worldwide
Used at all DOE labs

Exploring Tools (Ron Green et al.)

- Intel Studio XE 2016 Tools
- Vtune
- MKL
- Vector Advisor

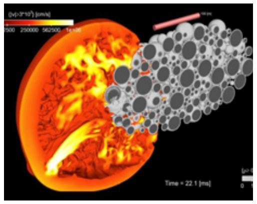
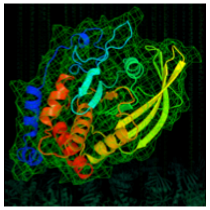
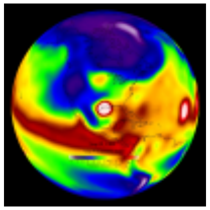
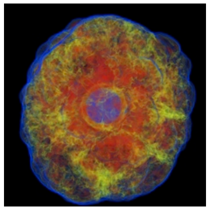
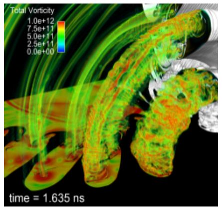
KNL Developer Challenges

- HBM – How apps take advantage?
  - Open question, answer will vary by application
  - Tools support Fortran:
    - !dir$ attributes fastmem :: <array or data object>
  - Tools Support C/C++: memkind API
    - https://github.com/memkind
  - Tools Support tracing – Intel Vtune support (coming)
- Hybrid Model MPI + X
  - Where X == Threading (OpenMP 4.x, pthreads, TBB, Cilk Plus)
- Vectorization – absolutely essential
  - -xmic-avx512 + Improved –qopt-report in Intel compilers v15 and v16
  - Intel Advisor XE – Vectorization Advisor tool
    - Premiers in the Intel Parallel Studio 2016 Aug 2015
    - Requires Intel compilers v16.0

Needs

- HBM Exploration
- Hackathons! (with real codes)
- White Boxes
- ???

Trinity Center of Excellence
LANL

Hai Ah Nam

August 2015

Trinity Center of What?

Center of Excellence refers to a team, a shared facility or an entity that provides leadership, best practices, research, support and/or training for a focus area.

PURPOSE: **"**Support the transition of key applications to the Trinity System"

Trinity Center of Who?

Center of Excellence refers to a **team, a shared facility or an entity** that provides leadership, best practices, research, support and/or training for **a focus area**.

PURPOSE: "Support the transition of key applications to the Trinity System"

Team: ***Cray*** – John Levesque (Senior Level Analyst); Mike Berry (on site); Jim Schwarzmeier; performance team
Intel – Ron Green (on site); monthly interaction with SMEs, Technical Advocates, and others as needed

Site: CoE Site Lead	Focus Area/Application
LANL: Tim Kelley	**xRage** (rad-hydro, multi-physics, multi-material, AMR)
SNL: Rob Hoekstra	Sierra (Fluid, Therm, Aero, Struct. Dynamics, Solid Mech.)
LLNL: Shawn Dawson	Mercury (MC neutron transport)

Trinity Center of Who?

Center of Excellence refers to a **team, a shared facility or an entity** that provides leadership, best practices, research, support and/or training for **a focus area**.

PURPOSE: "Support the transition of key applications to the Trinity System"

Team: ***Cray*** – John Levesque (Senior Analyst); Mike Berry (on site); Jim Schwarzmeier; performance team
Intel – Ron Green (on site); monthly interaction with SMEs, Technical Advocates, and others as needed
LANL – Tim Kelley (CCS-7), xRage development team, Hai Ah Nam (CCS-2), Ben Santos (HPC-3), Cornell Wright (HPC-5), Jorge Roman (HPC-3), Peter Lamborn (HPC-3)

Bi-weekly meeting (Tues @ 11AM, 200-116)

Trinity Center of Excellence

Center of Excellence refers to a team, a shared facility or an entity that provides **leadership, best practices, research,** support and/or training for a focus area.

PURPOSE: **"**Support the transition of key applications to the Trinity System"

Leadership, Best Practices, Research:

Early access to systems, software, tools
The first to bleed… uncover "features" that are in fact bugs

Engagement with other CoEs (Sierra, NERSC, ORNL, ANL) to bring back information to Trinity CoE & LANL

Share knowledge

1. hpc.lanl.gov → Platforms → Trinitite (Trinity)
2. ic-wiki.lanl.gov → Trinity Application Readiness (CoE, Open Science)

Trinity Center of Excellence

Center of Excellence refers to a team, a shared facility or an entity that provides leadership, best practices, research, **support and/or training** for a focus area.

PURPOSE: "Support the transition of **key applications** to the Trinity System"

Support & Training:

Cray Workshops & Tutorials

- 12 Steps to Performance Portability 6/16/15 (slides available)
- 1st 2 weeks December 2015 – Comprehensive Trinity Phase I Workshop
- 4-5 codes teams doing hands-on performance tutorials (contact if interested)
- 2 more KNL Workshops, dates TBD

Save the date. Agenda coming

Intel Discovery and Dungeon Sessions

- xRage Discovery Workshop (6/3)
- KNL Whitebox (early release)
- Deep dive on applications kernels

xRage on Trinity

- Major modernization efforts (cleaning house)
  - Break into packages
    - Clarify data & control flow, reduce complexity, separate concerns
  - Clean up within packages
    - Interfaces for packages, refactoring data structures
    - Long term goal: physics code is a stateless transformer
  - Migrate packages to target architecture
    - Prototyping ideas (tiled approach)
- Address identified bottlenecks & inhibitors of vectorization (e.g. solver, data structures)

Trinity Resources

- Trinity-coe-core@lanl.gov
  - Access to Cray & Intel SMEs
- LANL Help Resources
  - HPC (consult@lanl.gov)
  - Institutional Computing (ic-help@lanl.gov)
- Everyone
  - Sharing best practices (IC-wiki, HPC.lanl.gov)

Campaign Storage Backup Material

- **Managing Files**

We provide the following special parallel copy and list commands to efficiently manipulate and navigate Campaign Storage

- **pfcp**: Copy file(s) from *sourcePath* to *destinationPath in parallel*

gpfst1-fe$ pfcp fileA fileB gpfst1-fe$ pfcp -v fileA subdir gpfst1-fe$ pfcp -R subdirA subdirB*

- ***pfls**: Lists file(s) based on sourcePath in parallel*

gpfst1-fe$ pfls fileA gpfst1-fe$ pfls -v fileA gpfst1-fe$ pfls -R subdirA*

Visualization

Backup

Viz Tools/Software

- EnSight
  - Commercial tool supported by CEI
- ParaView
  - Open-source tool supported by Kitware Inc.
- VisIt
  - Open-source tool supported by LLNL

- All tools supported for Phase 1
- All tools support OpenSWR/Mesa for on-platform rendering (Haswell)

Viz Scenarios

- Interactive Visualization/Post-Processing
- In-situ Visualization
  - Tightly coupled simulation code and in-situ viz tool for run-time, in-memory visualization and analysis
  - Sierra/Catalyst (ParaView) & VisIt/LibSim supported
- In-transit Visualization
  - Visualization/analysis proceeds concurrently on other compute nodes
  - Potential use case for burst buffer
  - Active research area

Viz Models

- Off-platform rendering/Geometry Transfer
  - Interactive
  - Geometry/data is manipulated on Trinity viz nodes but rendered by viz tool on non-cluster hardware
- On-platform rendering/Image Transfer
  - Interactive
  - Images composited & rendered on Trinity viz nodes and sent to waiting viz tool
  - KNL rendering (with OpenSWR) is uncertain

Viz Nodes & Queue

- 2% of Haswell and Knights Landing partitions to be dedicated visualization/analysis queue
- Continues previous Cielo & ASC viz models
- Viz node allocation reviewed every ATCC cycle
- Additional visualization and analysis resources can be requested for special needs
- Nodes should support dynamic linking to shared object libraries and TCP/IP sockets for communication/DISCOM

Viz on Knights Landing

- Xeon Phi's have severe performance handicap when rendering through traditional OpenGL/ Mesa
  - Large impact on in-situ visualization
- Intel actively developing OpenSWR as a software rendering solution
  - OpenSWR integration with Mesa
  - First OpenSWR deployment to Haswell
  - Hopeful OpenSWR deployment for KNL
  - Tri-Lab, LBL and CEI/Kitware/VisIt development involvement in OpenSWR and KNL rendering

Sandia-specific viz

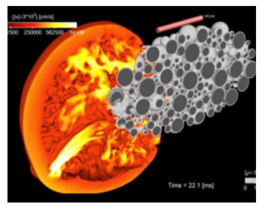
- EnSight
  - SNL: “ensight2cluster -m trinity”
- ParaView
  - SNL: “paraview2trinity”
- Sierra-Catalyst in-situ visualization
  - Existing capability supported as part of phase 1
  - Work is ongoing to support Aero

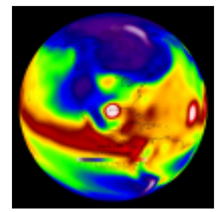
End of

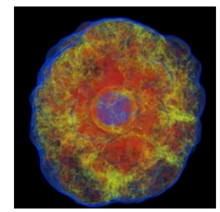
Visualization

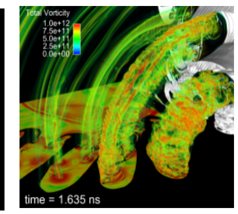
Backup

Trinity Filesystems
Begin Backup slides









Rob Hoekstra, Mike Glass, Ron Green(Intel) et al.

August 2015

SAND2015-6916 PE

RAID construction differs enabling different rebuild strategies

Traditional RAID6

- 8 Data + 2 Parity over 10 HDDs
  - Each 8+2 exists on 10 fixed HDDs
- One Disk Fails
  - Read 9 other parts from 9 HDDs
  - Write reconstructed part to one HDD
- Failed HDD must be replaced before reconstruction begins

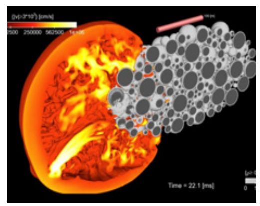
GridRAID

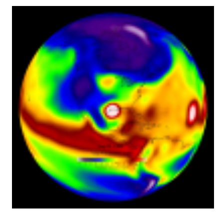
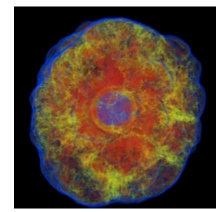
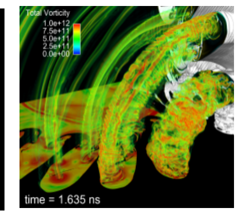
- 8 Data + 2 Parity over 41 HDDs
  - 2 spares distributed over 41 HDDs
- One Disk Fails
  - Read 9 other parts from 40 HDDs
  - Write reconstructed part over 40 HDDs
- Failed HDD must be replaced before chance of losing two more HDDs
  - Free-up spare space by writing spare parts from 40 HDDs to 1 HDD

GridRAID reconstructs ~4x faster than Traditional RAID

- Reconstruction time dominated by read
  - GridRAID reads from 40 HDDs vs. 9
- GridRAID replacement HDD reconstruction uses I/O load parameter
  - If PFS I/O load light, reconstruction uses more than minimum bandwidth setting
  - If PFS I/O load heavy, reconstruction won't exceed minimum bandwidth setting
    - Some performance degradation is possible

Trinity Filesystems
End Backup slides



Rob Hoekstra, Mike Glass, Ron Green(Intel) et al.

August 2015

SAND2015-6916 PE